PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G01V 3/10 — A1

(11) International Publication Number: WO 91/03746

(43) International Publication Date: 21 March 1991 (21.03.91)

(21) International Application Number: PCT/CA90/00286

(22) International Filing Date: 7 September 1990 (07.09.90)

(30) Priority data:
8920412.7 8 September 1989 (08.09.89) GB

(71) Applicant *(for all designated States except US)*: ATOMIC ENERGY OF CANADA LIMITED [CA/CA]; Chalk River, Ontario K0J 1J0 (CA).

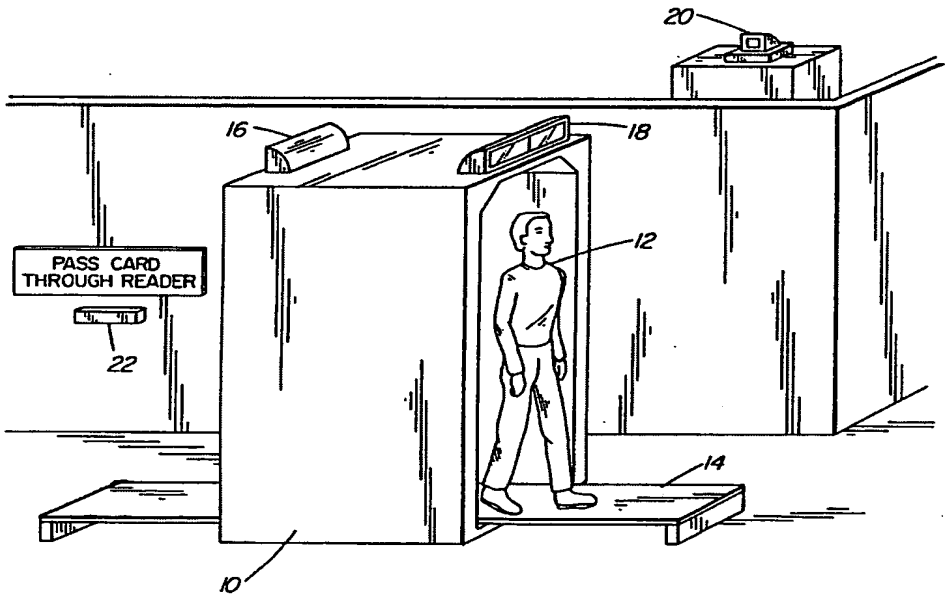
(72) Inventor; and
(75) Inventor/Applicant *(for US only)* : GHENT, Hugh, W. [CA/CA]; R.R. #1, Deep River, Ontario K0J 1P0 (CA).

(74) Agent: PROULX, Eugene, E.; Gowling, Strathy & Henderson, 160 Elgin Street, Suite 2600, Ottawa, Ontario K1N 8S3 (CA).

(81) Designated States: AT (European patent), AU, BE (European patent), CA, CH (European patent), DE (European patent)*, DK (European patent), ES (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent), US.

Published
With international search report.

(54) Title: METAL DETECTING SYSTEM



(57) Abstract

A metal detection station capable of detecting tiny metal pieces unlawfully concealed on persons moving through the station is disclosed. The station comprises a transmitting coil, receiving coils, field shaping coils, field shielding coils, shielding plates, receiver balancing coils, and a computer based console employing software to digitize, analyze, display, and store the detected signals.

* See back of page

DESIGNATIONS OF "DE"

Until further notice, any designation of "DE" in any international application whose international filing date is prior to October 3, 1990, shall have effect in the territory of the Federal Republic of Germany with the exception of the territory of the former German Democratic Republic.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MC	Monaco
AU	Australia	FI	Finland	MG	Madagascar
BB	Barbados	FR	France	ML	Mali
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Fasso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GR	Greece	NL	Netherlands
BJ	Benin	HU	Hungary	NO	Norway
BR	Brazil	IT	Italy	PL	Poland
CA	Canada	JP	Japan	RO	Romania
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SN	Senegal
CM	Cameroon	LI	Liechtenstein	SU	Soviet Union
DE	Germany	LK	Sri Lanka	TD	Chad
DK	Denmark	LU	Luxembourg	TG	Togo
				US	United States of America

METAL DETECTING SYSTEM

REFERENCE TO APPENDIX

A computer program used in accordance with the present invention is shown in the Appendix.

BACKGROUND OF THE INVENTION

The present invention relates, in general, to a metal detection system and, more specifically, to a method and an apparatus capable of detecting tiny metal pieces which may be unlawfully concealed on a person's body.

Industries involved with precious metals including precious metal refineries, mints, jewellery manufacturers and the like require a detecting apparatus which is capable of quickly and efficiently detecting small quantities of precious metals which may be unlawfully concealed on the person of an individual.

Many devices have been developed for detecting metals in the ground and elsewhere. However, such devices are intended to locate relatively massive concentrations of metal in static environments and, frequently, they are designed to exclude signals produced by small amounts of metal. For these and other reasons, metal detectors of this type are not suitable for screening individuals for concealed precious metals.

There also exists a number of metal detecting stations for detecting concealed firearms at airports, public offices, embassies, jails, and even hospitals. They are typically characterized by a portal unit which defines a passageway through which a person is required to pass. Electrical coils extend vertically in parallel relation around the portal unit with one coil, a transmitting coil, being energized by an alternating current to cause it to generate a magnetic field which in turn causes an electrical signal to be induced in the other coils, the receiving coils. Relatively massive metal objects, such as a gun, passed through the unit disturb the magnetic and hence the signals induced in the receive coils. Thus, the output of the coils are monitored to detect such disturbances. Heytow United States Patent No. 4,012,690 granted on March 15, 1977 and Mallick et al United States Patent No. 3,686,564 granted on August 22, 1972 describe devices of this type.

All the metal detecting stations, including those disclosed in the referenced patents, currently known to the inventor use electromagnetic principles. Generally, as indicated above, a transmitting coil connected to an electrical current driving unit is used to generate a time changing magnetic field and receiver coils are used to detect changes in the magnetic field. The changes in the magnetic field occur when either eddy currents are generated in conducting materials brought into the vicinity of the station or when ferromagnetic materials are brought near the station. Most ferrous material exhibit both effects - the eddy current and ferromagnetic effects. The extent to which the magnetic field is changed depends upon many factors including electrical conductivity, magnetic permeability, shape, size, structural properties, density, orientation and location relative to the transmitting coil. The human body is known to be a poor electrical conductor and has slight ferromagnetic properties. Due to the relatively large size of the human body a small but significant change occurs in the magnetic field of these stations when a person passes through. It has been found that, because the signal strength associated with a person is so much stronger than the signal strength associated with a small quantity of a precious metal, such stations are not inherently designed to detect the small quantities of metal as required in industries involved with precious metals including precious metal refineries, mints, jewellery manufacturers and the like.

Kerr United States Patent No. 4,719,421 granted on January 12, 1988 discloses a metal detector for detecting impurities in bakery products and comprises a transmit coil positioned between a pair of receive coils, as in the above referenced prior art, and circuitry which responds to deviation form a standard signal by generating a faulty product signal. The system can be set to distinguish between different types of loaves of bread. This arrangement suffers from the same difficulty mentioned earlier, that being that the size of the human body is so much larger than the small quantity of material it is difficult to distinguish signals associated with the small quantity of material from those associated with the individual carrying the material. Further, characteristics such as the size, shape, density and so forth of human body vary greatly from person to person and, accordingly, it is not possible to categorize humans in the manner

in which loaves of bread can be categorized, and to provide preset values as proposed by Kerr against which the output of the receive coils can be compared.

In summary, known metal detection devices do not specifically address the special problems associated with detecting concealed precious metal on humans or objects and to provide a permanent record of a violation when such is detected.

SUMMARY OF THE INVENTION

The principal object of this invention is to provide a means for detecting the smallest quantity of metal possible being transported, typically unlawfully and usually concealed, by persons passing through a detection station. It is a further object of the present invention to provide a detection system and process whereby the signal is obtained and can be displayed in a graphical form in order to demonstrate, in a court of law if necessary, the signal obtained from an individual passing through the metal detector.

The metal detector described hereafter has the capability of detecting significantly smaller quantities of metal carried by persons than any other currently known and available metal detector. It has the further advantage of displaying a recording of the actual signals obtained. It therefore is especially useful in industries dealing with precious metals including refineries, mints, jewellery manufacturers, etc. although use in a wide variety of other areas is also possible.

Accordingly in one aspect of the present invention there is provided a method of detecting concealed metals comprising the steps of generating a fluctuating electromagnetic field about a passageway, passing a person or object through the passageway, obtaining an electrical signal representative of the change of the electromagnetic field as the person or object passes through the passageway, comparing the values of predetermined characteristics of the electrical signal against the corresponding values of the characteristics of a base electrical signal for the person or object previously obtained when the person or object was known not to contain concealed metals, and generating an alarm signal when the differences between at least one of the characteristics signals exceeds a predetermined threshold.

In accordance with a further aspect of the present invention there is provided an apparatus for detecting concealed metals on a person or object, comprising means for generating an electromagnetic field about a passageway, means for producing an electrical signal representative of the change in the electromagnetic field as the person or object passes through the passageway, means for comparing the values of predetermined characteristics of the signal against corresponding values representative of an electrical calibration signal obtained when the person or object was known not to contain concealed metals, and means for generating an alarm signal when the differences between the values of at least one of the characteristics of the signals exceeds a predetermined threshold.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings wherein:

FIGURE 1 is a perspective view of a typical metal detecting station in accordance with a preferred embodiment of the present invention;

FIGURE 2 is a perspective view of the coil and winding configuration and associated circuitry for the detecting station;

FIGURE 3 is a diagrammatic electrical schematic of a circuit in accordance with a preferred embodiment of the present invention;

FIGURES 4a-4f illustrate the voltage signal changes in a receive coil as a typical metal-free human body passes through a detecting station with and without in the current generated by field-shaping coils;

FIGURES 5a-5c illustrate typical signal changes in the receive coil as various objects are brought close to the receive coil and then withdrawn;

FIGURE 6 is a logic flow chart illustrating the manner in which a person's calibration values are determined;

FIGURE 7 is a logic flow diagram illustrating the normal operation of the detection system.

DESCRIPTION OF PREFERRED EMBODIMENT

With reference now to the drawings, **FIGURE 1** shows a typical metal detection station comprising the open-ended enclosure or portal unit **10** with a person **12** walking through the portal unit on a shallow platform **14**, an entry display **16** and an exit display **18**, both mounted on the portal unit, a computer console **20** and a card reader **22** connected to the computer system. An electrical transmit coil, discussed later with reference to **FIGURES 2** and **3**, is disposed about the portal unit and generates a magnetic field at a predetermined alternating frequency. Similarly, electrical receive coils are disposed about the portal unit in a manner described later and produce alternating electrical signals in response the magnetic field in their vicinity. The signals are further processed and then digitized and transmitted to the computer system. The receive coils produce a characteristic signal when the portal is empty. Similarly, a person walking through the portal will influence the magnetic field in a distinctive manner and it has been found that it is possible to detect these distinctive characteristics from the output of the receive coils. Thus, in accordance with one aspect of the present invention, each person with pass privileges is required to walk through the portal without metal objects for the purpose of producing a base or calibration signal which is processed to quantify the characteristics. These characteristics are stored in computer memory so that whenever the individual passes through the portal unit, the computer will compute and compare the characteristics of the receive coil with the calibration characteristics to determine whether the individual is carrying concealed metals. It will be understood that the present invention is not limited to humans and that it is equally applicable to animals and inanimate objects. Thus, unless otherwise indicated, the reference to "person" is to be taken to apply to animals and inanimate objects.

Under normal operation, a person inserts an identification card into card reader **22** and waits for permission to walk through the portal unit. The card reader reads the card in known manner and transmits signals representative of the identity of the person to the computer system. If the person's status in the computer system is such that it is permissible for the person to pass through the portal unit, the display on the entry sign **16** will so indicate. The system is able

to recognize when the person has entered and exited the portal unit by comparing the output of the receive coils against the signal characteristics of an empty portal. When the system recognizes that the person has exited the portal, it processes the signals which were produced while the person was in the portal and compares them to the base or calibration values discussed earlier for that individual. Based upon this comparison, the exit display **18** will indicate either a "clean" or an "alarm" condition. The envelope of the signal generated (similar to that shown in **FIGURES 4** and **5**) when the person passed through the portal can be displayed graphically on the computer monitor for visual observation by the operator. This display and other relevant data may be printed on paper to serve as a permanent record if desired.

FIGURE 2 illustrates an arrangement of coils in accordance to the preferred embodiment of the present invention. A transmit coil **30** is disposed substantially midway along the length of the portal and is operable to produce an alternating magnetic field within the portal. A pair of receive coils **32** are coaxially disposed on opposite sides and equidistantly from the transmit coil and operate to produce an electrical signal in response to changes to magnetic fields within their respective fields of influence. A field-shaping coil **34** is coaxially disposed proximate each receive coil and on the side thereof remote from the transmit coil. The field-shaping coils serve to reduce the effect of objects, such as a person's feet, placed in close proximity to the receive coils. Finally, at least one pair of field-shielding coils **36** are coaxially disposed on the axially outer sides of the field-shaping coils. These coils serve to reduce the magnetic field extending axially from the portal unit. While the field-shaping and field-shielding coils enhance system performance, they are not essential to successful operation of the system. All of the coils are located in planes which are perpendicular to the axis of the passageway through portal unit **10** and are fixedly mounted in or on the walls of the portal unit to avoid vibrations which would disturb the magnetic fields by suitable insulated mountings (not shown). The portal unit also incorporates magnetic field shielding comprising thin narrow strips of ferromagnetic material in at least its top and side walls as described in more detail later.

With reference to **FIGURE 3**, a driver **40** delivers a high frequency current to transmit coil **30** at a frequency, typically 25 kHz, determined by oscillator **42**. This rapidly changing current generates a magnetic field about the axis of transmit coil. A capacitor **44** is connected across the input leads to the transmit coil and is chosen to cause a resonant condition in the transmit coil and serves to significantly amplify the current in the transmit coil and the magnetic field strength about the coil.

The receive coils are responsive, by producing a current therein, to all magnetic fields and changing currents in their respective vicinities including the alternating magnetic field produced by the transmit coil, the field-shaping coils, field-shielding coils and ferromagnetic material and ferrous material in which eddy currents are produced. If the coils are wound with precision, such that the spacings between respective windings are accurately controlled and substantially identical on each side of the transmit coil, then the currents and voltages induced in each receive coil will be substantially identical.

As best shown in **FIGURE 3**, the receive coils **32** are connected together in series such that as the voltage in one coil increases, the voltage in the other decreases. In this way, the resultant voltage across both coils is substantially zero. A capacitor **46** across the output leads of the receive coils is chosen such that it will cause a resonant condition between the two receive coils. Any small voltage differences between the two windings will then be greatly increased due to this resonant condition.

In practice, it is virtually impossible to wind the transmit and receive coils with exact precision. These differences, along with general wiring connections and other external influences, will cause significant non-symmetrical influences on the magnetic fields inside the portal unit. Therefore, the induced voltages in each receive coil are not in general identical - albeit they are very close. Hence, it is necessary, in order to achieve a balanced condition, to provide a means to adjust or trim the voltage induced in each receive coil. A quadrature adjusting means is provided to balance the "in-phase" and the "out-of-phase" components of the signals. There are in general many conceivable ways to accomplish this adjustment. In the illustrated embodiment, a wire loop **47** having its axis normal

to the axis of the coils, and miscellaneous adjustments of the transmit leads are used to zero the induced voltage across the combined receive coils.

The field-shaping coils are used to generate an additional magnetic field in such a way as to enhance the detection of metal. It will be noticed that the field-shaping coils **34** are in close proximity to the receive coils **32**. It should also be noted that, typically, the only part of the human body that comes in close proximity to the receive coils are the feet (typically, a few centimetres spacing). Due to physical constraints, it is impractical to design units in which this is not the case. The design of the units can be such that greater clearances for arms, hands and heads are achieved. The voltage change in the receive coils due to a particular object increases as the distance from the object to the receive coil decreases. Therefore, the receive coils are inherently more sensitive to the feet of an individual walking through the unit. Furthermore, significantly greater variation in the signal received by the receive coil will occur depending on the foot position relative to the receive coils. This type of variation will not be as significant for other parts of the body. In order to reduce the detrimental effects of the variation in the voltage induced in the receive coils due to the position of the feet, an additional current can be generated in the field-shaping coils. The advantages of having this additional current will become apparent later. In the present configuration, a separate phase shifter **48** and driver **49** are used to control the phase and amplitude of the current in the field shaping coils. An alternate means of achieving similar currents in the field-shaping coils could be achieved by carefully designing the coil and selecting appropriate passive components (resistors, capacitors and inductors) and not require the use of the phase shifter **48** and driver **49**.

The primary function of the field shielding coils **36** is to effectively reduce the magnetic field extending axially from the portal unit. The induced current in these coils creates a magnetic field which opposes (or reduces) the magnetic field which generates it. In this way, the magnetic field extending from the portal unit is also decreased. Therefore, ferromagnetic materials, which tend to focus the magnetic field through these materials when brought into close proximity to the front or rear of the unit, will have a greatly reduced influence on the

non-symmetrical magnetic field distribution generated inside the portal unit. In the present configuration, the field shielding coils are shown to be located or spaced a distance axially outwardly from the respective transmit and receive coils. This is necessary in order to minimize the effect the field-shielding coils would have on the magnetic field in the vicinity of the receive coil.

The previously noted narrow ferromagnetic shielding strips (not shown) extend over the exterior surface of the portal unit on at least the sides and top. The purpose of these strips is to constrain, as much as possible, the magnetic fields generated by the transmit and field-shaping coils so they remain substantially within the portal unit. If no significant magnetic fields are able to leave the unit, then no significant eddy currents can be induced in electrically conducting materials exterior to the unit, and, if no eddy currents are generated, then no additional magnetic fields can be generated which would normally have been sensed by the receive windings. Similarly, ferromagnetic components close to the exterior of the unit would have a minimal influence on the magnetic fields inside the unit. This unit therefore provides a degree of electromagnetic shielding to the front and sides.

The output of the receive coils are connected to detection electronics commonly found in commercially-available, eddy current non-destructive testing instrumentation. One type of eddy current instrument which has been used successfully in laboratory tests is known as the "Defectomat F", by Institut. Dr. Forster of West Germany. Less expensive eddy current instruments are available from a number of suppliers. Other instruments are available from the following companies: Eddy Current Technology, Elotest, Hocking, Nortec; Reluxtrol, Tecrad and Zetec, certain of which can supply separate digitizing cards. This electronics, in general, has the capability of effectively balancing the small residual voltage remaining on the receive coil after the walk-through unit has been installed, and detecting the change in magnitude and direction (phase) of the voltage change in the receive coils. As explained below, the output of the detection electronics is comprised of two voltages which can be resolved into a resistive voltage change and a reactive voltage change. During system operation,

these voltages are being constantly digitized by separate digital circuitry, and the values are received by the computers.

FIGURE 3 illustrates the basic components and operation of the detection electronics. The output terminals of the receive coils are each connected to an input of an amplifier **50** whose output **52** is connected to one input terminal **54** of a summer or adder **56**. The output of oscillator **42** is passed through an attenuator **58** and a phase shifter **60** and applied to the other input terminal **62** of the summer. The output of the summer is a signal which is the difference of the two input signals. Thus, when the portal is empty, the output of the receiver coils will be the same as the input to the transmit coil and, accordingly, the difference in the magnitude of the two signals will be zero. However, if a person or object is passing through the portal, the magnitude and phase of the output of the receiver coils will be different, resulting in a non-zero summer output. The summer output is delivered to a demodulator **66** which produces two DC level signals, one being representative of the resistive component of the signal and the other being representative of the reactive component of the signal. These two signals are then delivered to an analog-to-digital converter which digitizes the signals and feeds the digitized signals to the computer **70.**

Before describing the digital signal processing scheme of the present invention, it would be useful to briefly review the types of signals sensed by the receive coils. An understanding of these signals is required in order to appreciate the value of field shaping coils **34** and to understand how the signal analysis is able to discriminate differences in the processed receive coil voltages.

As indicated above, the induced signal in the receive coils can be resolved into two components, namely, a resistive component and a reactive component. These components can be plotted on a graph in which the reactive component is plotted along the ordinate (y-axis) and the resistive component is plotted along the abscissa (x-axis). When a ferromagnetic material is brought into the vicinity of the receive coil, the magnetic field inside the coil increases due to the higher magnetic permeability of the ferromagnetic material. The increase in magnetic field strength causes a voltage increase in the receive winding (**FIGURE 4**). The voltage increase is 90° out-of-phase with the voltage induced in the receive coil

due to the magnetic field generated by the current in the transmit coil, and is therefore identified as a voltage increase having a direction along the positive reactive axis (see Signal No. 1 in **FIGURE 4**). Now consider the effects on the detection system of two loops of copper wire, each about 5 cm in diameter. One loop will be made of a wire having a very small cross-sectional area (for example #43 AWG) and the other of a larger cross-sectional area (for example #14 AWG). When the thin wire loop is brought close to one of the receive coils, a voltage will be induced in it due to the magnetic field generated from the current in the transmit coil. Because the resistance of the thin loop is high relative to the loop's reactance, the current in the loop will be substantially in phase with the induced voltage. As a result the magnetic field due to the current in the thin loop will increase, and hence the voltage induced in the receive coil will increase with an "in-phase" orientation (see Signal No. 2 as shown in **FIGURE 4**). When the thicker wire loop is brought close to the receive coil, a similar voltage is induced in it from the magnetic field generated by the currents in the transmit coil. However, the reactive impedance is far greater than the resistive impedance of this loop, and, as a result, the current generated in the loop generates a magnetic field which opposes the magnetic field which induced the voltage. As a result, the voltage induced in the receive coil will decrease, and this decrease is 90° out-of-phase with the initial voltage induced in the receive coil from the transmit coil (see Signal No. 3 in **FIGURE 4**).

Therefore, it can be understood from the above examples that, depending upon an object's magnetic and conductive properties, shape, size, structural properties, density, orientation and location relative to the various coils in the metal detector, a range of signals having orientations of up to 180° is possible. Typically, most precious metals have a high conductivity and so the voltage change tends to be in the negative reactive direction. However, very thin materials (for example platinum gauze) will exhibit properties like that of the thin wire.

It will be seen that while a person or object passes through the portal unit the magnetic field about each receive coil will fluctuate. If the receive coil output is sampled at equal increments of time, resolved into its resistive and reactive components and plotted on a graph as described above, a curve similar to that

illustrated in **FIGURES 4** and **5** will result. The curve is characterized by two loops **80** and **82** in which loop **80** indicates the response of the receive coil proximate the entrance of the portal unit and loop **82** indicates the response of the receive coil proximate the exit of the portal unit. Each loop has an approach section **84** in which the strength of the signal increases in magnitude and a departure section **86** in which the strength of the signal decreases in magnitude.

As stated earlier, the human body is known to be a poor conductor and has slight ferromagnetic properties. Due to the size of the body, the signal change detected by the receive coils due to the presence of the body can be significantly greater than the signal of interest, typical of the small metal object being carried by the person. A typical signal, sensed by the receive coils from the human body as it passes through the metal detector being described, is shown in **FIGURE 5a.** This signal shows the response of each receive coil, and hence we see the positive and negative components of the signal. It is clear from this signal that the body has a much higher conductive component than it has a ferromagnetic component. **FIGURE 5b** shows the much smaller response for a tiny metal object alone as it is passed through the portal unit and **FIGURE 5c** shows the signal when the tiny metal object is being passed through the detector on the human body. It follows from the foregoing that the output of the receive coils will vary with the manner in which an individual walks through the unit, i.e., side to side and front to back motion, swinging arms, etc. These variations cause the normal signal to vary somewhat and potentially inhibit the ability to detect small quantities of metal. To increase the detectability of small objects, it is therefore advisable to require employees to walk through the unit in a reasonable consistent and predetermined way.

The effect of the field shaping coils can now be appreciated. In order to demonstrate the effect of a person's foot as he walks through the portal, signals from a plastic container containing a salt water solution were obtained. This container was moved through the portal in a manner similar to that of a person's foot, and the signals are shown in **FIGURES 4a** and **4b.** It will be noted that the signal without the field shaping current is approximately 50% greater than the signal with the field shaping current. **FIGURES 4c** and **4e** show the signal from

a person walking through the portal without (**FIGURE 4c**) and with (**FIGURE 4e**) the field shaping current. The signal without the field shaping current is only about 20% greater than that with the field shaping current. This is expected since the predominant effect of the field shaping current will occur on things closest to it (i.e. the feet) and to a lesser extent on things more removed (i.e. the legs, torso and head). **FIGURE 4d** and **4f** show the signals of a person walking through the portal with a coin in the front pocket without (**FIGURE 4d**) and with (**FIGURE 4f**) the field shaping current. The change in the middle angle is significantly greater for the case with the field shaping current (about 13°) than without the field shaping current (about 9°). Therefore, by utilizing field shaping currents, a significant increase in the detectability of small metal objects being carried by individuals is obtained.

Each person who walks through the portal unit must have first been calibrated. The calibration process is illustrated in the flow chart of **FIGURE 6.** The calibration process consists of the individual walking through the unit a number of times - for example 8 times. During each pass through the unit, the computer system receives the digitized voltage values. Based on these values, it determines when the signal starts and ends, and computes various characteristics of the signal for each pass. These characteristics can include, for example, the maximum positive and negative reactive components, the maximum positive and negative resistive components, the sum of the absolute values of the positive and negative reactive components of the signal, the sum of the absolute values of the positive and negative resistive components of the signal, the angle defined by these two sums called the differential angle, the angle of the middle portion, (e.g. the middle three-fifths) of each of the approach and departure sections of the two loops, and a middle angle defined by the terminal portion of the departure section of the first loop and the initial portion of the approach section of the second loop. These angles may be computed by any suitable technique, such as linear regression techniques, well known to those skilled in the art. The mean and standard deviation of each of these characteristics are then computed, and can be checked against the individual's weight, height and other significant features (e.g., implants, dental bridges etc.) to ensure the values are consistent with those

expected. If the values are satisfactory, they are stored in computer memory in association with other particulars, such as name, identification code and the like, for the individual. These values are considered the individual's calibration values and are used during normal operation. An advantage of this approach is that it considerably reduces the amount of computer memory required for each individual. It is possible, of course, to store the raw data, samples taken at small increments of time as the person passes through the portal unit. However, such an approach would consume a considerable amount of computer memory and would introduce considerable complexity into the signal analysis algorithm.

During normal operation, as illustrated by the logic flow chart of **FIGURE 7**, after the individual has been identified by the computer system and instructed to walk through the unit, the computer system receives the processed receive coil voltage changes in digitized form, determines the characteristic values of the signal and then compares these values to the calibration values previously described. Based on this comparison, a decision is made as to whether an alarm or non-alarm condition is present. The results of this decision are then displayed on exit display **18** so that an attending security guard can take appropriate action.

APPENDIX

```
'  Program PMD is designed to operate a precious metal detector.
'  The program requires a relay card to operate controls used to indicate
'  the outcome of a pass through the detector, and a separate card used to
'  set signal levels from the detector and to digitive the signals.

DECLARE FUNCTION TCGAINSET% CDECL (BYVAL GAIN%)
DECLARE FUNCTION TCFREQSET& CDECL (BYVAL FREQ&, BYVAL PHASE%)
DECLARE SUB TCACQUISITION CDECL
DECLARE FUNCTION TCCOMPENSATION% CDECL
DECLARE SUB TCINITIALIZATION CDECL
DECLARE SUB MAIN.MENU.DISPLAY ()
DECLARE SUB GETFEATURES (condition%)
DECLARE SUB GETSIGNATURE ()
DECLARE SUB GETINKEY ()
DECLARE SUB UPDATE.EMPLOYEE.DATABASE () .
DECLARE SUB DISPLAY.DATABASE ()
DECLARE SUB PMDCONS (A%)
DECLARE SUB PMDHEAD ()
DECLARE SUB GET.DATA ()
DECLARE SUB COMPENSATE ()
DECLARE SUB OPERATE ()
DECLARE SUB BLANKLINE (LINE.TO.BLANK%)
DECLARE SUB ADDNEWCHAR ()
DECLARE SUB READ.KEYBOARD ()
DECLARE SUB CHECK.ID ()
DECLARE SUB DELETE.EMPLOYEE.FROM.DATABASE ()
DECLARE SUB GET.DAYS.FROM.1990 ()
DECLARE SUB SWAP.EMPLOYEE (HIGH.INDEX.NUM%, ARRAY.ENTRY.POINT%)
DECLARE SUB COPY.EMPLOYEE.DATA (OLD%, NEW%)

COMMON A$, EXIT$, FINISH$, IPRM%, IPRMMAX%, COLUMN%, ASPACECHR$
COMMON TIMELOOP%, R%, C%
COMMON XAVG%, YAVG%, XREF%, YREF%, X.AT.COMP%, Y.AT.COMP%
COMMON COMPCHECK%, COMP#()
COMMON NUM.UPDATE.HEAD%, UPDATEHEADR%(), UPDATEHEADC%(), UPDATEHEADT$()
COMMON     NUM.UPDATE.ADD.HEAD%,     UPDATEADDHEADR%(),     UPDATEADDHEADC%(),
UPDATEADDHEADT$()
COMMON NUM.UPDATE.PARM%, UPDATEROW%(), UPDATECOL%(), UPDATESPCS%()
COMMON DATAR%(), DATAC%(), SPCS%()
COMMON NUMBER.OF.EMPLOYEES%, DATA.CHANGE$
COMMON DMEAN(), DSTDEV(), NUM.FEATURES%
COMMON   DACTLAST$(),   DACTFIRST$(),   DACTINIT$(),   DCARDLAST$(),   DCARDNO&(),
DEMPNO%()
COMMON DSEX() AS STRING * 1, DHEIGHT(), DWEIGHT()
COMMON DBRIDGE() AS STRING * 1, DIMPLANT() AS STRING * 1, DPACE() AS STRING *
1, DOTHER$()
COMMON IN.SECURE.AREA%(), ANTIPASS1%, ANTIPASS2%, READER.OPERATIONAL$
COMMON PERSON.CALIBRATED$, ALARM$, OPERATE.ALERT%()
COMMON PERSON.IN.DATABASE$, RUN.MODE$
COMMON EMPNO%, INDEX.NUM%
COMMON INTERRUPT.RATE, RISETIME, DIFFERENTIAL.TIME, MAX.WALK.TIME%
COMMON MAX.SAMPLES.FOR.EVENT%, SAMPLES.IN.RISETIME%
```

```
COMMON M$(), REFLINE, PI, RADTODEG, LOOPSPERSEC, BLINKTIME
COMMON PDISO%, PDISO.WAIT%, PDISO.GO%, PDISO.ALARM%, PDISO.CLEAN%
COMMON CLEAN.TIME%, ALARM.TIME%, PERCENT.ALARM%
COMMON MAX.PASSNUM%, NUM.OPERATE.ALERTS%
COMMON OP.NSTDEV(), CAL.NSTDEV(), LIM.STDEV()
COMMON RECOMP.CHECK%, FREQ&, PHASE%, GAIN%, DRANGE%
COMMON FILE.CONSTANTS$, LAST.DATE$, LAST.TIME$
COMMON X.DEL.CHK.6%, Y.DEL.CHK.S%, X.DEL.CHK.E%, Y.DEL.CHK.E%
COMMON CALIBRATE.MAX.DELAY.TIME%, OPERATE.MAX.DELAY.TIME%
COMMON VIEW.LENGTH, LBND%, UBND%
COMMON CONPARM$(), CALQUEST$()
COMMON NUM.CAL.HEAD%, CALHEADR%(), CALHEADC%(), CALHEADT$()
COMMON NUM.CON.HEAD%, CONHEADR%(), CONHEADC%(), CONHEADT$()
COMMON NUM.CAL.PARM%, CALROW%(), CALCOL%(), CALSPCS%()
COMMON NUM.CON.PARM%, CONROW%(), CONCOL%(), CONSPCS%()
COMMON E%, NEWFREQ&, NEWGAIN%, NALARM%, ALARMNUM%()
COMMON FEATURE(), X%(), Y%(), NSAMP%
COMMON COUNT%, IRT%, IHT%, XAR%(), YAR%(), A%()
COMMON NAME.FILE$, ERRNUM%, REASON$
COMMON YES$, NO$, OK$, NOTOK$
COMMON PERSON.WAITING$, NEED.TO.BLANK$
COMMON F1HIT$, LASTKEY$, PASSNUM%
COMMON MAX.SAMPLES.BEFORE.EVENT%, PASS.ALERT%, END.EVENT%
COMMON DM%(), DAYS.FROM.1990%

CLEAR , , 2000
  DIM A%(0 TO 4), COMP#(3)
  DIM OPERATE.ALERT%(10), ALARMNUM%(10)
  DIM FEATURE(10)
  DIM HEADR%(25), HEADC%(25), HEADT$(25), SPCS%(25)
  DIM DATAR%(25), DATAC%(25), DATASPCS%(25)
  DIM CALHEADR%(25), CALHEADC%(25), CALHEADT$(25)
  DIM CALROW%(25), CALCOL%(25), CALSPCS%(25), CALPARM$(25)
  DIM CONHEADR%(25), CONHEADC%(25), CONHEADT$(25)
  DIM CONROW%(25), CONCOL%(25), CONSPCS%(25), CONPARM$(25)
  DIM UPDATEHEADR%(25), UPDATEHEADC%(25), UPDATEHEADT$(25)
  DIM UPDATEADDHEADR%(25), UPDATEADDHEADC%(25), UPDATEADDHEADT$(25)
  DIM UPDATEROW%(25), UPDATECOL%(25), UPDATESPCS%(25), UPDATEPARM$(25)
  DIM CALQUEST$(25), M$(12), DM%(12)
  NMAX% = 50
  DIM DEMPNO%(NMAX%), DACTLAST$(NMAX%)
  DIM DACTFIRST$(NMAX%), DACTINIT$(NMAX%)
  DIM DSEX(NMAX%) AS STRING * 1, DHEIGHT(NMAX%), DWEIGHT(NMAX%)
  DIM DBRIDGE(NMAX%) AS STRING * 1, DIMPLANT(NMAX%) AS STRING * 1
  DIM DPACE(NMAX%) AS STRING * 1, DOTHER$(NMAX%)
  DIM DMEAN(NMAX%, 10), DSTDEV(NMAX%, 10)
  DIM OP.NSTDEV(10), CAL.NSTDEV(10), LIM.STDEV(10)

  CALL PMDCONS(0)

  DIM XAR%(MAX.SAMPLES.FOR.EVENT%), YAR%(MAX.SAMPLES.FOR.EVENT%)
  DIM X%(MAX.SAMPLES.FOR.EVENT%), Y%(MAX.SAMPLES.FOR.EVENT%)
```

```
DIM XCALDAT%(8, MAX.SAMPLES.FOR.EVENT%), YCALDAT%(8, MAX.SAMPLES.FOR.EVENT%)
DIM NCALDAT%(8)

TIME.TEST% = 0: JEND% = 1: IEND% = 20000
WHILE TIME.TEST% = 0
  JEND% = 2 * JEND%
  STTIME = TIMER
  FOR J% = 1 TO JEND%: FOR I% = 1 TO IEND%: NEXT I%: NEXT J%
  ENDTIME = TIMER
  IF ENDTIME - STTIME > 1.5 THEN TIME.TEST% = 1
WEND
LOOPSPERSEC = JEND% / (ENDTIME - STTIME) * IEND%
BLINKTIME = .05
TIMELOOP% = BLINKTIME * LOOPSPERSEC

YES$ = "YES": NO$ = "NO": OK$ = "OK": NOTOK$ = "NOT OK"
STAT.BASE% = 40' - the number used for determining the mean and st. dev.
SB% = STAT.BASE%: SBM% = STAT.BASE% - 1
ESCKEY$ = CHR$(27)
ADAY% = VAL(MID$(DATE$, 4, 2))
MONTH$ = M$(VAL(LEFT$(DATE$, 2)))
RANDOMIZE TIMER
X.DEL.CHK.INV.S% = 2 * X.DEL.CHK.S%
Y.DEL.CHK.INV.S% = 2 * Y.DEL.CHK.S%
SCREEN 0, , 1, 1: CLS
GOSUB EMPLOYEE.DATABASE
ASPACECHR$ = "_": SPACECHR$ = STRING$(80, ASPACECHR$)
DRIVE.ERROR$ = NO$
CALL PMDHEAD
GOSUB SET.BOARD
CALL MAIN.MENU.DISPLAY
RESPONSER% = CSRLIN: RESPONSEC% = POS(0)
 Note that the initial value for RESPONSE$="O" so that the OPERATE mode
 starts right away.
RESPONSE$ = "O"
WHILE RESPONSE$ <> "-Q"
  SELECT CASE RESPONSE$
    CASE "C"
      GOSUB CALIBRATE
      CALL MAIN.MENU.DISPLAY
      RESPONSE$ = ""
    CASE IS = "O"
      RUN.MODE$ = "OPERATE"
      IF DATA.CHANGE$ = YES$ THEN GOSUB SAVE.DATABASE
      GOSUB OPERATE
      CALL MAIN.MENU.DISPLAY
      RESPONSE$ = ""
    CASE "U"
      CALL UPDATE.EMPLOYEE.DATABASE
      CALL MAIN.MENU.DISPLAY
      RESPONSE$ = ""
    CASE "D"
```

```
        CALL DISPLAY.DATABASE
        CALL MAIN.MENU.DISPLAY
        RESPONSE$ = ""
      CASE IS = "Q"
        LOCATE 23, 1: PRINT "STOP"
      CASE ELSE
        LOCATE RESPONSER% + 2, 1
        SOUND 500, 2: SOUND 300, 2
        PRINT "Invalid entry - try again";
      END SELECT
    SOUND 100, 3: SOUND 50, 2
    LOCATE RESPONSER%, RESPONSEC%
    CALL GETINKEY
    RESPONSE$ = UCASE$(A$)
    PRINT A$;
    WEND
  END

CALIBRATE: ' - routine to calibrate the individual
  RUN.MODE$ = "CALIBRATE": OKAY.TO.CALIBRATE$ = YES$
  SCREEN 10, , 0, 0: CLS : ESCHIT$ = NO$
  MAX.SAMPLES.BEFORE.EVENT% = CALIBRATE.MAX.DELAY.TIME% * INTERRUPT.RATE
  COLOR 3, 0:
  LOCATE 1, 1: PRINT CHR$(201) + STRING$(78, 205) + CHR$(187); : LOCATE 2, 1
  PRINT CHR$(186) + SPACE$(31) + "CALIBRATION MODE" + SPACE$(31) + CHR$(186)
  LOCATE 3, 1: PRINT CHR$(200) + STRING$(78, 205) + CHR$(188);
  COLOR 1, 0

' - need to get the employee's last name and their employee number.
  GET.NAME$ = YES$
  WHILE GET.NAME$ = YES$ AND ESCHIT$ = NO$
    CALL BLANKLINE(8): PRINT "Enter the employee's last name ";
    GOSUB GET.INPUT: LASTNAME$ = ASTR$
    IF LEN(LASTNAME$) > 0 THEN LASTNAME$ = LTRIM$(RTRIM$(UCASE$(LASTNAME$)))
    CALL BLANKLINE(9): PRINT "Enter the employee's number ";
    GOSUB GET.INPUT: EMPSTR$ = ASTR$: EMPNO% = VAL(ASTR$): CALEMPNO% = EMPNO%
    CALL CHECK.ID: ' - returns PERSON.IN.DATABASE$, PERSON.CALIBRATED$, and
                       INDEX.NUM%
    CAL.INDEX.NUM% = INDEX.NUM%
    IF PERSON.IN.DATABASE$ = NO$ THEN
      CALL BLANKLINE(13): CALL BLANKLINE(12): CALL BLANKLINE(11)
      PRINT "    The employee number "; EMPSTR$; " is not in the database."
      PRINT "          Do you wish to re-enter the name and number (Y/N) ? ";
      SROWQ% = CSRLIN: SCOLQ% = POS(0)
      GOSUB CHECK.INKEY: PRINT A$; : A$ = UCASE$(A$)
      WHILE A$ <> "Y" AND A$ <> "N"
        CALL BLANKLINE(14): PRINT "Invalid entry - try again "
        LOCATE SROWQ%, SCOLQ%: GOSUB CHECK.INKEY: PRINT A$: A$ = UCASE$(A$)
      WEND
      IF A$ = "N" THEN GET.NAME$ = NO$: OKAY.TO.CALIBRATE$ = NO$
    ELSE
' - now to make sure the last names match
```

```
    NAME.MATCH$ = YES$
    IF DACTLAST$(CAL.INDEX.NUM%) <> LASTNAME$ THEN
      NAME.MATCH$ = NO$
      I% = CAL.INDEX.NUM%
      CALL BLANKLINE(13): CALL BLANKLINE(12): CALL BLANKLINE(11)
      PRINT "   Employee "; DACTFIRST$(I%); " "; DACTINIT$(I%); " ";
      PRINT DACTLAST$(I%); " has employee number "; EMPNO%
      PRINT "          The last names are not the same."
      PRINT "       Do you wish to re-enter the name and number (Y/N) ? ";
      SROWQ% = CSRLIN: SCOLQ% = POS(0)
      GOSUB CHECK.INKEY: PRINT A$; : A$ = UCASE$(A$)
      WHILE A$ <> "Y" AND A$ <> "N"
        CALL BLANKLINE(15): PRINT "Invalid entry - try again "
        LOCATE SROWQ%, SCOLQ%: GOSUB CHECK.INKEY: PRINT A$: A$ = UCASE$(A$)
      WEND
      IF A$ = "N" THEN GET.NAME$ = NO$: OKAY.TO.CALIBRATE$ = NO$
    END IF
    IF NAME.MATCH$ = YES$ THEN
      GET.NAME$ = NO$
      IF PERSON.CALIBRATED$ = YES$ THEN
        CALTYPE$ = "R"
      ELSE
        CALTYPE$ = "N"
      END IF
    END IF
  END IF
WEND
IF OKAY.TO.CALIBRATE$ = YES$ AND ESCHIT$ = NO$ THEN
  GOSUB GET.CAL.DATA
  GOSUB CALIBRATE.OLD.HEAD.DISPLAY
  IF CALTYPE$ = "N" THEN
    GOSUB CALIBRATE.DISPLAY.CHANGE
  ELSE
    CALL BLANKLINE(18)
    PRINT "Do you wish to change any values - enter   Y(yes)   N(no) ";
    ACAL$ = ""
    WHILE ACAL$ <> "Y" AND ACAL$ <> "N" AND ACAL$ <> CHR$(27)
      SOUND 500, 2: SOUND 300, 2: LOCATE 18, 61
      GOSUB CHECK.INKEY: PRINT A$; : ACAL$ = UCASE$(A$)
      SELECT CASE ACAL$
        CASE "Y"
          CALL BLANKLINE(18): CALL BLANKLINE(20)
          GOSUB CALIBRATE.DISPLAY.CHANGE
        CASE "N"
          GOSUB CALIBRATION.SETUP.NAME
        CASE CHR$(27)
          OKAY.TO.CALIBRATE$ = NO$
        CASE ELSE
          LOCATE 20, 5: PRINT "Invalid response - try again.";
          LOCATE 18, 61: PRINT " ";
      END SELECT
    WEND
```

```
    END IF
  END IF
' - Note that in routine CALIBRATE.DISPLAY.CHANGE the parameter
'       OKAY.TO.CALIBRATE$ = NO$ if the escape key is pressed.
  IF OKAY.TO.CALIBRATE$ = YES$ AND ESCHIT$ = NO$ THEN
    PASSNUM% = 1: PASS.ALERT% = 0: PASS1CHECK$ = "NOT OK"
    VIEW PRINT 13 TO 24
    CLS 2
    PRINT "PASS        DELX    DELY DIF.AN MID.AN   ANGL1   ANGL2   ANGR1   ANGR2
DELYL  DELYR"
    WHILE PASSNUM% <= MAX.PASSNUM% AND PASS.ALERT% = 0 AND ESCHIT$ = NO$
      PASS.ALERT% = 0
      CALL COMPENSATE
      CALL BLANKLINE(25): SOUND 500, 2: SOUND 200, 2
      PRINT "Ok to walk through - elbows against sides,";
      PRINT " hands on middle - PASS "; PASSNUM%;
      IF ESCHIT$ = NO$ THEN
        IF RELAY.CONTROLS$ = YES$ THEN OUT PDISO%, PDISO.GO%
        GOSUB PLOT.CAL.BORDER
        CALL GETSIGNATURE
        FOR I% = 1 TO NSAMP%
          XCALDAT%(PASSNUM%, I%) = X%(I%)
          YCALDAT%(PASSNUM%, I%) = Y%(I%)
        NEXT I%
        NCALDAT%(PASSNUM%) = NSAMP%
        IF END.EVENT% > 0 THEN
          CALL GETFEATURES(0)
          LOCATE 13 + PASSNUM%, 1
          PRINT USING "###"; PASSNUM%; : PRINT "     ";
          FOR I% = 1 TO NUM.FEATURES%
            CALFEATURE(PASSNUM%, I%) = FEATURE(I%)
            PRINT USING "#####.#"; FEATURE(I%);
          NEXT I%: PRINT
        END IF
        EVENT% = 0: COUNT% = -1: TOTAL.COUNT% = -1: OLDTIME = TIMER
        IRT% = COUNT% - SAMPLES.IN.RISETIME%
        IHT% = COUNT% - SAMPLES.IN.RISETIME% / 2
        PASSNUM% = PASSNUM% + 1
      END IF
    WEND
' - now I need to permit additional passes in order to satisfy the
'   CHECK.STATS conditions.
    IF PASS.ALERT% = 0 AND ESCHIT$ = NO$ THEN
      REDO.PASS% = -1
      WHILE REDO.PASS% <> 0 AND PASS.ALERT% = 0
        GOSUB GET.STATS: GOSUB PRINT.STATS
        STATS.OK$ = YES$: GOSUB CHECK.STATS
        IF REDO.PASS% <> 0 THEN
          PASSNUM% = REDO.PASS%: PASS.ALERT% = 0
          CALL BLANKLINE(25): SOUND 500, 2: SOUND 200, 2
          CALL COMPENSATE
          PRINT "Ok to walk through - elbows against sides,";
```

```
            PRINT " hands on middle - RE-PASS "; PASSNUM%;
            OUT PDISO%, PDISO.GO%
            GOSUB PLOT.CAL.BORDER
            CALL GETSIGNATURE
            FOR I% = 1 TO NSAMP%
              XCALDAT%(PASSNUM%, I%) = X%(I%)
              YCALDAT%(PASSNUM%, I%) = Y%(I%)
            NEXT I%
            NCALDAT%(PASSNUM%) = NSAMP%
            IF END.EVENT% > 0 THEN
              CALL GETFEATURES(0)
              LOCATE 13 + PASSNUM%, 1
              PRINT USING "###"; PASSNUM%; : PRINT "    ";
              FOR I% = 1 TO NUM.FEATURES%
                CALFEATURE(PASSNUM%, I%) = FEATURE(I%)
                PRINT USING "#####.#"; FEATURE(I%);
              NEXT I%: PRINT
            END IF
            EVENT% = 0: COUNT% = -1: TOTAL.COUNT% = -1: OLDTIME = TIMER
            IRT% = COUNT% - SAMPLES.IN.RISETIME%
            IHT% = COUNT% - SAMPLES.IN.RISETIME% / 2
          END IF
        WEND
        IF STATS.OK$ = YES$ THEN '          - okay to update employees info
          I% = CAL.INDEX.NUM%
          DSEX(I%) = CALEMPSEX$: DHEIGHT(I%) = CALEMPHEIGHT
          DWEIGHT(I%) = CALEMPWEIGHT: DBRIDGE(I%) = CALEMPBRIDGE$
          DIMPLANT(I%) = CALEMPIMPLANT$: DPACE(I%) = CALEMPPACE$
          DOTHER$(I%) = CALEMPOTHER$
' - now I need to determine which pass is the most representative and to save
'   the data on the hard drive.
          SUMMIN = 1000000: PASS.SAVE% = 1
          FOR J% = 1 TO MAX.PASSNUM%
            SUM = 0
            FOR K% = 1 TO NUM.FEATURES%
            SUM = SUM + ABS(CALFEATURE(J%, K%) - DMEAN(I%, K%)) / DSTDEV(I%, K%)
            NEXT K%
            IF SUM < SUMMIN THEN SUMMIN = SUM: PASS.SAVE% = J%
          NEXT J%
          AFN$ = "E" + LTRIM$(RTRIM$(STR$(DEMPNO%(I%))))
          DRIVE.B.ERROR$ = YES$
          ON ERROR GOTO DISCB.CHECK
          CAL.FILE.NAME$ = "B:" + AFN$
          WHILE DRIVE.B.ERROR$ = YES$
            DRIVE.B.ERROR$ = NO$
            OPEN CAL.FILE.NAME$ FOR OUTPUT AS #1
          WEND
          GOSUB SAVE.CAL.DATA
          ON ERROR GOTO 0
          CAL.FILE.NAME$ = "C:\PMD\CALDAT\" + AFN$
          OPEN CAL.FILE.NAME$ FOR OUTPUT AS #1
          GOSUB SAVE.CAL.DATA
```

```
        CALL BLANKLINE(25): CALL BLANKLINE(24): BEEP
        PRINT "Calibration was successful - ";
        PRINT "Press return to continue. ";
        GOSUB CHECK.INKEY
        DATA.CHANGE$ = YES$
      ELSE
        FOR I% = 1 TO NUM.FEATURES%
          DMEAN(INDEX.NUM%, I%) = 0: DSTDEV(INDEX.NUM%, I%) = 0
        NEXT I%
        CALL BLANKLINE(25): CALL BLANKLINE(24): BEEP
        PRINT "Calibration values are not consistent -";
        PRINT "press return to continue. ";
        GOSUB CHECK.INKEY
      END IF
    ELSE
      CALL BLANKLINE(25): CALL BLANKLINE(24): BEEP
      PRINT "Problem encountered while collecting data - ";
      PRINT "press return to continue. ";
      CALL GETINKEY
    END IF
  END IF
  ALARM$ = NO$
  RETURN

SAVE.CAL.DATA: ' - routine to save the calibration data
  I% = CAL.INDEX.NUM%
  WRITE #1, DATE$, TIME$, PASS.SAVE%
  A1$ = DACTLAST$(I%)
  A2$ = DACTFIRST$(I%): A3$ = DACTINIT$(I%): A4% = DEMPNO%(I%)
  A5$ = DSEX(I%)
  A6 = DHEIGHT(I%): A7 = DWEIGHT(I%): A8$ = DBRIDGE(I%)
  A9$ = DIMPLANT(I%): AA$ = DPACE(I%): AB$ = DOTHER$(I%)
  WRITE #1, A1$, A2$, A3$, A4%, A5$, A6, A7, A8$, A9$, AA$, AB$
  B1 = DMEAN(I%, 1): B2 = DMEAN(I%, 2): B3 = DMEAN(I%, 3)
  B4 = DMEAN(I%, 4): B5 = DMEAN(I%, 5): B6 = DMEAN(I%, 6)
  B7 = DMEAN(I%, 7): B8 = DMEAN(I%, 8): B9 = DMEAN(I%, 9)
  BA = DMEAN(I%, 10)
  WRITE #1, B1, B2, B3, B4, B5, B6, B7, B8, B9, BA
  B1 = DSTDEV(I%, 1): B2 = DSTDEV(I%, 2): B3 = DSTDEV(I%, 3)
  B4 = DSTDEV(I%, 4): B5 = DSTDEV(I%, 5): B6 = DSTDEV(I%, 6)
  B7 = DSTDEV(I%, 7): B8 = DSTDEV(I%, 8): B9 = DSTDEV(I%, 9)
  BA = DSTDEV(I%, 10)
  WRITE #1, B1, B2, B3, B4, B5, B6, B7, B8, B9, BA
  FOR J% = 1 TO NUM.FEATURES%
    A1 = CALFEATURE(1, J%): A2 = CALFEATURE(2, J%): A3 = CALFEATURE(3, J%)
    A4 = CALFEATURE(4, J%): A5 = CALFEATURE(5, J%): A6 = CALFEATURE(6, J%)
    A7 = CALFEATURE(7, J%): A8 = CALFEATURE(8, J%)
    WRITE #1, A1, A2, A3, A4, A5, A6, A7, A8
  NEXT J%
  WRITE #1, NCALDAT%(PASS.SAVE%)
  FOR J% = 1 TO NCALDAT%(PASS.SAVE%)
    WRITE #1, XCALDAT%(PASS.SAVE%, J%), YCALDAT%(PASS.SAVE%, J%)
```

```
    NEXT J%
    CLOSE #1
    RETURN

DISCB.CHECK: ' - routine to check that drive B is okay to use
    PRINT "Please insure a formatted disc (which isn't full) is in drive B."
    PRINT "Press return to continue ";
    GOSUB CHECK.INKEY
    DRIVE.B.ERROR$ = YES$
    RESUME NEXT

GET.EMP.NAME: ' - routine to get an employee name and employee number
    GET.NAME$ = YES$
    WHILE GET.NAME$ = YES$
      CALL BLANKLINE(8): PRINT "Enter the employee's last name ";
      GOSUB GET.INPUT: LASTNAME$ = ASTR$
      IF LEN(LASTNAME$) > 0 THEN LASTNAME$ = LTRIM$(RTRIM$(UCASE$(LASTNAME$)))
      CALL BLANKLINE(9): PRINT "Enter the employee's number ";
      GOSUB GET.INPUT: EMPSTR$ = ASTR$: EMPNO% = VAL(LEFT$(ASTR$, 4))
      CALEMPNO% = EMPNO%
      OLD.MODE$ = RUN.MODE$
      RUN.MODE$ = "CALIBRATE"
      CALL CHECK.ID: ' - returns PERSON.IN.DATABASE$, PERSON.CALIBRATED$, and
                       INDEX.NUM%
      RUN.MODE$ = OLD.MODE$
      CAL.INDEX.NUM% = INDEX.NUM%
      IF PERSON.IN.DATABASE$ = NO$ THEN
        CALL BLANKLINE(13): CALL BLANKLINE(12): CALL BLANKLINE(11)
        PRINT "    The employee number "; EMPSTR$; " is not in the database."
        PRINT "          Do you wish to re-enter the name and number (Y/N) ? ";
        RPOS% = CSRLIN: CPOS% = POS(0)
        GOSUB CHECK.INKEY: PRINT A$; : A$ = UCASE$(A$)
        WHILE A$ <> "Y" AND A$ <> "N"
          CALL BLANKLINE(14): PRINT "Invalid entry - try again "
          LOCATE RPOS%, CPOS%: GOSUB CHECK.INKEY: PRINT A$: A$ = UCASE$(A$)
        WEND
        IF A$ = "N" THEN GET.NAME$ = NO$: OKAY.TO.CALIBRATE$ = NO$
      ELSE
' - now to make sure the last names match
        NAME.MATCH$ = YES$
        IF DACTLAST$(INDEX.NUM%) <> LASTNAME$ THEN
          NAME.MATCH$ = NO$
          I% = INDEX.NUM%
          CALL BLANKLINE(13): CALL BLANKLINE(12): CALL BLANKLINE(11)
          PRINT "    Employee "; DACTFIRST$(I%); " "; DACTINIT$(I%); " ";
          PRINT DACTLAST$(I%); " has employee number "; EMPNO%
          PRINT "                The last names are not the same."
          PRINT "          Do you wish to re-enter the name and number (Y/N) ? ";
          GOSUB CHECK.INKEY: PRINT A$; : A$ = UCASE$(A$)
          WHILE A$ <> "Y" AND A$ <> "N"
            CALL BLANKLINE(15): PRINT "Invalid entry - try again "
            LOCATE 13, 70: GOSUB CHECK.INKEY: PRINT A$: A$ = UCASE$(A$)
```

```
        WEND
        IF A$ = "N" THEN GET.NAME$ = NO$: OKAY.TO.CALIBRATE$ = NO$
      END IF
      IF NAME.MATCH$ = YES$ THEN
        GET.NAME$ = NO$
        IF PERSON.CALIBRATED$ = YES$ THEN
          CALTYPE$ = "R"
        ELSE
          CALTYPE$ = "N"
        END IF
      END IF
    END IF
  WEND
  RETURN

CALIBRATE.OLD.HEAD.DISPLAY: ' - routine to display the CALIBRATE screen
  CLS (0)
  COLOR 3, 0
  FOR I% = 1 TO NUM.CAL.HEAD%
    LOCATE CALHEADR%(I%), CALHEADC%(I%)
    PRINT CALHEADT$(I%);
  NEXT I%
  COLOR 1, 0
  FOR I% = 1 TO NUM.CAL.PARM%
    LOCATE CALROW%(I%), CALCOL%(I%): PRINT LEFT$(SPACECHR$, CALSPCS%(I%));
    LOCATE CALROW%(I%), CALCOL%(I%): PRINT CALPARM$(I%);
  NEXT I%
  RETURN

CALIBRATE.DISPLAY.CHANGE: ' - routine to change the entries on the calibration
                            display.
  IPRMSTART% = 5: IPRMMAX% = NUM.CAL.PARM%: IPRM% = IPRMSTART%
  FOR I% = 1 TO IPRMMAX%
    DATAR%(I%) = CALROW%(I%): DATAC%(I%) = CALCOL%(I%): SPCS%(I%) = CALSPCS%(I%)
  NEXT I%
  ALLDATA$ = "NOT OK"
  WHILE ALLDATA$ = "NOT OK"
    FINISH$ = NO$
    WHILE FINISH$ = NO$
      IF IPRM% > IPRMMAX% THEN IPRM% = IPRMSTART%
      IF IPRM% = 1 THEN IPRM% = IPRMSTART%
      IF IPRM% < IPRMSTART% THEN IPRM% = IPRMMAX%
      CALL BLANKLINE(25): PRINT CALQUEST$(IPRM%); : SOUND 100, 3: SOUND 50, 2
      ROW% = DATAR%(IPRM%): COLUMN% = DATAC%(IPRM%)
      LOCATE ROW%, COLUMN%
      GOSUB ADDNEWCHAR.WITH.CHECK
      IF EXIT$ = YES$ THEN
        OKAY.TO.CALIBRATE$ = NO$
        RETURN
      END IF
      CALL BLANKLINE(15)
    WEND
```

```
    GOSUB CALIBRATION.SETUP.NAME
    GOSUB CALIBRATION.SETUP.CHECK
  WEND
  RETURN

ADDNEWCHAR.WITH.CHECK: '  routine to add a character
'                        as well as excessive signals in the PMD.
  A$ = ""
  EXIT$ = NO$
  WHILE A$ <> CHR$(13)
    GOSUB CHECK.INKEY
    SELECT CASE ASC(A$)
      CASE 13: ' - to advance to the next field
        IPRM% = IPRM% + 1
        IF IPRM% > IPRMMAX% THEN IPRM% = 1
      CASE 32 TO 217: ' - standard characters
        PRINT A$; : A$ = "": C% = POS(0)
        IF (C% - COLUMN%) >= SPCS%(IPRM%) OR C% < COLUMN% THEN
          IPRM% = IPRM% + 1: A$ = CHR$(13)
          IF IPRM% > IPRMMAX% THEN IPRM% = 1
        END IF
      CASE 8: ' - backspace and overwrite
        LOCATE R%, C%: PRINT ASPACECHR$
        LOCATE R%, C% - 1: A$ = "": C% = POS(0)
        IF C% < COLUMN% THEN LOCATE R%, COLUMN%
      CASE 9: ' - tab to the next field
        IPRM% = IPRM% + 1
        IF IPRM% > IPRMMAX% THEN IPRM% = 1
        A$ = CHR$(13)
      CASE 27: ' - escape key - used to exit data entry - no checking
        A$ = CHR$(13)
        EXIT$ = YES$
      CASE 0: ' - look at keys having 2 byte responses
        SELECT CASE ASC(RIGHT$(A$, 1))
          CASE 15: ' - shift tab - go back a field
            IPRM% = IPRM% - 1
            IF IPRM% <= 0 THEN IPRM% = IPRMMAX%
            A$ = CHR$(13)
          CASE 75: ' - shift left 1 character
            LOCATE R%, C% - 1
            A$ = "": C% = POS(0)
            IF C% < COLUMN% THEN LOCATE R%, COLUMN%
          CASE 77: ' - shift right 1 character
            CP% = C% + 1
            IF CP% <= 80 THEN LOCATE R%, CP%: A$ = "": C% = POS(0)
            IF (C% - COLUMN%) >= SPCS%(IPRM%) OR C% = 80 THEN
              IPRM% = IPRM% + 1: A$ = CHR$(13)
              IF IPRM% > IPRMMAX% THEN IPRM% = 1
            END IF
          CASE 68: ' - F10 - exit menu
            FINISH$ = YES$
            A$ = CHR$(13)
```

```
            CASE ELSE
              A$ = ""
          END SELECT
        CASE ELSE
      END SELECT
    WEND
    RETURN

CALIBRATION.SETUP.NAME: '    -     subroutine to name the PAGE parameters
    FOR I% = 1 TO NUM.CAL.PARM%
      B$ = " "
      FOR J% = 1 TO CALSPCS%(I%)
        A$ = CHR$(SCREEN(CALROW%(I%), CALCOL%(I%) + J% - 1))
        IF A$ = ASPACECHR$ THEN
          J% = CALSPCS%(I%) + 1
        ELSE
          B$ = B$ + A$
          END IF
        NEXT J%
      CALPARM$(I%) = RTRIM$(LTRIM$(B$))
      NEXT I%
    CALLASTNAME$ = CALPARM$(1)
    CALFIRSTNAME$ = CALPARM$(2)
    CALINITIAL$ = CALPARM$(3)
    CALEMPNUMBER& = VAL(CALPARM$(4))
    CALEMPSEX$ = CALPARM$(5)
    CALEMPHEIGHT = VAL(CALPARM$(6))
    CALEMPWEIGHT = VAL(CALPARM$(7))
    CALEMPBRIDGE$ = CALPARM$(8)
    CALEMPIMPLANT$ = CALPARM$(9)
    CALEMPPACE$ = CALPARM$(10)
    CALEMPOTHER$ = CALPARM$(11)
    RETURN

CALIBRATION.SETUP.CHECK: ' - subroutine to check the data entered is okay
    ALLDATA$ = "OK"
    FOR I% = IPRMSTART% TO NUM.CAL.PARM%
      IF LEN(CALPARM$(I%)) < 1 THEN
        ALLDATA$ = "NOT OK"
        CALL BLANKLINE(15): PRINT "A value must be entered here."
        SOUND 500, 2: SOUND 300, 2
        IPRM% = I%
        I% = NUM.CAL.PARM% + 1
        END IF
      NEXT I%
    IF ALLDATA$ = "OK" AND (CALEMPHEIGHT < 100 OR CALEMPHEIGHT > 220) THEN
      ALLDATA$ = "NOT OK": IPRM% = 6
      CALL BLANKLINE(15): PRINT "Correct the person's height value"
      SOUND 500, 2: SOUND 300, 2
      END IF
    IF ALLDATA$ = "OK" AND (CALEMPWEIGHT < 30 OR CALEMPWEIGHT > 200) THEN
      ALLDATA$ = "NOT OK": IPRM% = 7
```

```
      CALL BLANKLINE(15): PRINT "Correct the person's weight value "
      SOUND 500, 2: SOUND 300, 2
      END IF
      A$ = CALEMPSEX$
    IF ALLDATA$ = "OK" AND A$ <> "M" AND A$ <> "m" AND A$ <> "F" AND A$ <> "f"
 THEN
      ALLDATA$ = "NOT OK": IPRM% = 5
      CALL BLANKLINE(15): PRINT "NOTE : a 'M' or 'F' must be entered here."
      SOUND 500, 2: SOUND 300, 2
      END IF
    IF ALLDATA$ = "OK" THEN
      FOR I% = 8 TO 10
        A$ = CALPARM$(I%)
        IF A$ <> "N" AND A$ <> "n" AND A$ <> "Y" AND A$ <> "y" THEN
          ALLDATA$ = "NOT OK"
          CALL BLANKLINE(15): PRINT "NOTE : a 'N' or 'Y' must be entered here."
          SOUND 500, 2: SOUND 300, 2
          IPRM% = I%
          I% = NUM.CAL.PARM% + 1
          END IF
        NEXT I%
      END IF
    RETURN

CHECK.INKEY: ' - This subroutine gets a character from the keyboard using the
'     INKEY command, and also checks the PMD for an invalid entry.
    A$ = ""
    WHILE A$ = ""
      R% = CSRLIN: C% = POS(0): ACHAR = SCREEN(R%, C%): PRINT " ";
      IF ACHAR <= 32 THEN ACHAR = ASC(ASPACECHR$)
      LOCATE R%, C%
      FOR I% = 1 TO TIMELOOP%: NEXT I%: PRINT CHR$(ACHAR);
      FOR I% = 1 TO TIMELOOP%: NEXT I%: LOCATE R%, C%
      A$ = INKEY$
      IF A$ = ESCKEY$ THEN ESCHIT$ = YES$
      GOSUB CHECK.INVALID.ENTRY: ' - need to check for excessive signal level.
    WEND
    RETURN

GET.INPUT: ' - This subprogram gets an input from the keyboard using the
'     GETINKEY subprogram, and also checks the PMD for an invalid entry.
    ASTR$ = "": A$ = "": CREF% = POS(0): RREF% = CSRLIN: C% = CREF%: R% = RREF%
    PERSON.WAITING$ = NO$
    WHILE A$ <> CHR$(13) AND PERSON.WAITING$ = NO$ AND A$ <> CHR$(27)
      GOSUB CHECK.INKEY
      SELECT CASE ASC(A$)
        CASE 32 TO 217
          ASTR$ = ASTR$ + A$
          PRINT A$;
          C% = C% + 1
        CASE 8: ' - backspace character
          IF LEN(ASTR$) > 1 THEN
```

```
            ASTR$ = LEFT$(ASTR$, LEN(ASTR$) - 1)
            LOCATE R%, C% - 1: PRINT " ": LOCATE R%, C% - 1
          ELSE
            ASTR$ = ""
            LOCATE RREF%, CREF%: PRINT " ": LOCATE RREF%, CREF%
          END IF
        CASE 27: ' - for the ESC key
          ESCHIT$ = "YES"
        CASE ELSE
      END SELECT
    WEND
    RETURN

GET.CAL.DATA: ' - routine to get the calibration data for the employee
    E% = INDEX.NUM%
    CALPARM$(1) = DACTLAST$(E%): CALPARM$(2) = DACTFIRST$(E%)
    CALPARM$(3) = DACTINIT$(E%): CALPARM$(4) = RTRIM$(LTRIM$(STR$(DEMPNO%(E%))))
    CALPARM$(5) = DSEX(E%)
    CALPARM$(6) = RTRIM$(LTRIM$(STR$(DHEIGHT(E%))))
    CALPARM$(7) = RTRIM$(LTRIM$(STR$(DWEIGHT(E%))))
    CALPARM$(8) = DBRIDGE(E%)
    CALPARM$(9) = DIMPLANT(E%): CALPARM$(10) = DPACE(E%)
    CALPARM$(11) = DOTHER$(E%)
    RETURN

GET.STATS: ' - routine to get the mean and standard deviation for feature
    FOR I% = 1 TO NUM.FEATURES%
      SUM = 0
      FOR J% = 1 TO MAX.PASSNUM%: SUM = SUM + CALFEATURE(J%, I%): NEXT J%
      AMEAN = SUM / MAX.PASSNUM%
      SUM = 0
      FOR J% = 1 TO MAX.PASSNUM%
        SUM = SUM + (CALFEATURE(J%, I%) - AMEAN) ^ 2
        NEXT J%
      ASTDEV = SQR(SUM / MAX.PASSNUM%)
      IF ASTDEV < LIM.STDEV(I%) THEN ASTDEV = LIM.STDEV(I%)
      DSTDEV(INDEX.NUM%, I%) = ASTDEV
      DMEAN(INDEX.NUM%, I%) = AMEAN
      NEXT I%
    RETURN

PRINT.STATS: ' - routine to print the mean and standard deviations calculated.
    LOCATE 13 + MAX.PASSNUM% + 1, 1: PRINT "MEAN    ";
    FOR I% = 1 TO NUM.FEATURES%
      PRINT USING "#####.#"; DMEAN(INDEX.NUM%, I%);
    NEXT I%: PRINT
    LOCATE 13 + MAX.PASSNUM% + 2, 1: PRINT "ST.DEV ";
    FOR I% = 1 TO NUM.FEATURES%
      PRINT USING "#####.#"; DSTDEV(INDEX.NUM%, I%);
    NEXT I%: PRINT
    RETURN
```

```
CHECK.STATS: ' - routine to determine the mean and standard deviations
'                are consistent with the employee's data.
' - first, the values must all fall within 2 standard deviations
  I% = 1: REDO.PASS% = 0
  WHILE I% <= NUM.FEATURES% AND REDO.PASS% = 0
    HIVAL = DMEAN(INDEX.NUM%, I%) + CAL.NSTDEV(I%) * DSTDEV(INDEX.NUM%, I%)
    LOVAL = DMEAN(INDEX.NUM%, I%) - CAL.NSTDEV(I%) * DSTDEV(INDEX.NUM%, I%)
    PASS% = 1
    WHILE PASS% <= MAX.PASSNUM% AND REDO.PASS% = 0
      IF CALFEATURE(PASS%, I%) > HIVAL OR CALFEATURE(PASS%, I%) < LOVAL THEN
        REDO.PASS% = PASS%: CHECK.FEATURE% = I%: STATS.OK$ = NO$
        CALL BLANKLINE(25)
        PRINT "Feature "; CHECK.FEATURE%; " on Pass "; REDO.PASS%;
        PRINT ; " isn't ok - press return. ";
        GOSUB CHECK.INKEY
        END IF
      PASS% = PASS% + 1
      WEND
    I% = I% + 1
    WEND
' - now to ensure the data is consistent with the employee height,weight and
'   other information.

  RETURN

PLOT.CAL.BORDER: ' - routine to plot the border for the calibration plots.
  DPVV% = 83: DPVH% = 100
  SPACE.H% = 139 + DPVH% * (PASSNUM% - 4 * INT((PASSNUM% - 1) / 4))
  SPACE.V% = 0 + DPVV% * INT((PASSNUM% - 1) / 4)
  VIEW (SPACE.H%, SPACE.V%)-(SPACE.H% + DPVH%, SPACE.V% + DPVV%)
  WINDOW (LBND%, LBND%)-(UBND%, UBND%)
  CLS 1
  PSET (LBND%, LBND%)
  LINE -(LBND%, UBND%): LINE -(UBND%, UBND%): LINE -(UBND%, LBND%)
  LINE -(LBND%, LBND%)
  RETURN

OPERATE: ' - routine to operate the PMD in normal operation.
  MAX.SAMPLES.BEFORE.EVENT% = OPERATE.MAX.DELAY.TIME% * INTERRUPT.RATE
  ALARM.ON$ = NO$: NEED.TO.BLANK$ = YES$
  SCREEN 10, , 1, 1: CLS 0: COLOR 1
  LOCATE 1, 1: PRINT CHR$(201) + STRING$(27, 205) + CHR$(187);
  LOCATE 2, 1: PRINT CHR$(186) + SPACE$(27) + CHR$(186);
  LOCATE 3, 1: PRINT CHR$(186) + SPACE$(27) + CHR$(186);
  LOCATE 4, 1: PRINT CHR$(200) + STRING$(27, 205) + CHR$(188);
  COLOR 3
  LOCATE 2, 5: PRINT "THE WISE COMPANY'S"
  LOCATE 3, 3: PRINT "PRECIOUS METAL DETECTOR"
  LOCATE 5, 8: PRINT "OPERATION MODE": COLOR 1
  LASTKEY$ = "": F1HIT$ = NO$
  EVENT% = 0: COUNT% = -1: TOTAL.COUNT% = -1: OLDTIME = TIMER
  IRT% = COUNT% - SAMPLES.IN.RISETIME%: IHT% = COUNT% - SAMPLES.IN.RISETIME% /
```

```
2
  WHILE F1HIT$ = NO$
'  In the OPERATE mode an employee number is entered from the keyboard.
    EMPNO% = 0
    CALL READ.KEYBOARD: ' - returns EMPNO%
    IF EMPNO% > 0 THEN PERSON.WAITING$ = YES$
    IF PERSON.WAITING$ = YES$ THEN
      GOSUB PROCESS.PERSON.WAITING
    ELSE
' - situation when there is nobody waiting to walk through the PMD.
      GOSUB CHECK.INVALID.ENTRY
    END IF
    GOSUB CHECK.KEYBOARD: ' - returns F1HIT$
    ESCHIT$ = NO$
    IF A$ = CHR$(4) THEN ' - CNTL T presèed for changing the time
      OP.ROW% = CSRLIN: OP.COL% = POS(0)
      SCREEN 10, , 0, 0
      SHELL "DATE": SHELL "TIME"
      SCREEN 10, , 1, 1: LOCATE OP.ROW%, OP.COL%
    END IF
  WEND
  RETURN


EMP.DATA.SAVE.ERROR: ' - routine to check for Drive B
  LPRINT : LPRINT CHR$(27) + "W1"; : LPRINT CHR$(27) + "w1";
  LPRINT "EMPLOYEE DATA IS NOT SAVED"
  LPRINT CHR$(27) + "W0"; : LPRINT CHR$(27) + "w0"
  DRIVE.ERROR$ = YES$
  RESUME NEXT

PROCESS.PERSON.WAITING: ' - routine to control the procedure for the person
'    waiting to pass through the PMD.

  FOR I% = 1 TO NUM.OPERATE.ALERTS%: OPERATE.ALERT%(I%) = 0: NEXT I%
  PERSON.WAITING$ = YES$
  CALL CHECK.ID: ' routine returns  PERSON.CALIBRATED$, ALARM$
'                 and PERSON.IN.DATABASE$

' - If a person is waiting to go through the PMD, then the 'GO' light will
'   be activated, and the person is expected to go through the unit.  The
'   ALARM.ON$ = YES$ occurs when excessive noise is detected.  When this
'   condition exists, access to the portal is denied until the noise goes
'   away.
  WHILE ALARM.ON$ = YES$
    GOSUB CHECK.INVALID.ENTRY
  WEND

'   - For the person who is in the database and is calibrated, access to the
'     PMD is granted, and the persons signal is digitized and processed.
  PASS.ALERT% = 0
  E% = INDEX.NUM%
```

```
    IF INDEX.NUM% > 0 THEN
      AINIT$ = DACTINIT$(E%)
      ANAME$ = DACTFIRST$(E%) + " " + LEFT$(AINIT$, 1) + " " + DACTLAST$(E%)
    ELSE
      ANAME$ = " UN-KNOWN "
    END IF
    GOSUB DEFINE.OPERATE.BORDER
    GOSUB PLOT.OPERATE.BORDER: ' - selects SCREEN 10,,1,1(page 1)
    LOCATE 8, 1: PRINT SPACE$(30): LOCATE 8, 1: PRINT ANAME$;
    OP.COL% = POS(0) + 2: OP.ROW% = 8
    CALL GET.DATA
    IF ABS(XAVG% - X.AT.COMP%) > RECOMP.CHECK% THEN
      CALL COMPENSATE
    ELSE
      IF ABS(YAVG% - Y.AT.COMP%) > RECOMP.CHECK% THEN CALL COMPENSATE
    END IF
    GOSUB RELAY.WALK
    CALL GETSIGNATURE
    IF PASS.ALERT% = 0 THEN
      CALL GETFEATURES(0)
      IF PASS.ALERT% = 0 THEN GOSUB COMPARE.FEATURES
    END IF
    PERSON.CLEAN$ = YES$
    IF ALARM$ = YES$ THEN
      PERSON.CLEAN$ = NO$
    ELSE
      GOSUB RANDOM.ALARM
    END IF
    IF ALARM$ = YES$ THEN
      GOSUB RELAY.ALARM
    ELSE
      LPRINT MONTH$; " "; MID$(DATE$, 4, 2); " at "; TIME$; " ";
      LPRINT "Employee "; ANAME$; "  clean."
      OUT PDISO%, PDISO.WAIT% + PDISO.CLEAN%
      ON TIMER(CLEAN.TIME%) GOSUB CLEAN.OFF
      TIMER ON
    END IF

' - now I need to update the mean and standard deviation values used.
    IF PERSON.CLEAN$ = "YES" THEN
      E% = INDEX.NUM%
      FOR I% = 1 TO NUM.FEATURES%
        DMEAN(E%, I%) = (DMEAN(E%, I%) * SBM% + FEATURE(I%)) / SB%
        ADEL = ABS(FEATURE(I%) - DMEAN(E%, I%))
        IF ADEL > LIM.STDEV(I%) THEN
          IF ADEL < DSTDEV(E%, I%) OR ADEL < 1.5 * LIM.STDEV(I%) THEN
            NEW.STDEV = (DSTDEV(E%, I%) * SBM% + ADEL ^ 2) / SB%
          END IF
        END IF
      NEXT I%
    END IF
```

```
    LOCATE OP.ROW%, OP.COL%
    IF ALARM$ = YES$ THEN
      COLOR 2: PRINT " alarmed.": COLOR 1
    ELSE
      PRINT " clean."
    END IF

    PERSON.WAITING$ = NO$
    EVENT% = 0: COUNT% = -1: TOTAL.COUNT% = -1: OLDTIME = TIMER
    IRT% = COUNT% - SAMPLES.IN.RISETIME%
    IHT% = COUNT% - SAMPLES.IN.RISETIME% / 2
    RETURN

DEFINE.OPERATE.BORDER: ' - routine to define the border for the operate mode.
    DOTS.PER.VIEW.V% = VIEW.LENGTH / 13.9 * 350
    DOTS.PER.VIEW.H% = VIEW.LENGTH / 20.4 * 640
    SPACE.H% = 639 - DOTS.PER.VIEW.H%
    VIEW (SPACE.H%, 1)-(SPACE.H% + DOTS.PER.VIEW.H%, 1 + DOTS.PER.VIEW.V%)
    WINDOW (LBND%, LBND%)-(UBND%, UBND%)
    CLS 1
    RETURN

PLOT.OPERATE.BORDER: ' - routine to plot the border
    SCREEN 10, , 1, 1
    VIEW PRINT 7 TO 24: CLS 2
    VIEW (SPACE.H%, 1)-(SPACE.H% + DOTS.PER.VIEW.H%, 1 + DOTS.PER.VIEW.V%)
    WINDOW (LBND%, LBND%)-(UBND%, UBND%)
    CLS 1
    PSET (LBND%, LBND%)
    LINE -(LBND%, UBND%): LINE -(UBND%, UBND%): LINE -(UBND%, LBND%)
    LINE -(LBND%, LBND%)
    RETURN


COMPARE.FEATURES: ' - routine to compare signature features with cal. features.
    E% = INDEX.NUM%: NALARM% = 0
    FOR I% = 1 TO NUM.FEATURES%
      IF FEATURE(I%) < DMEAN(E%, I%) - OP.NSTDEV(I%) * DSTDEV(E%, I%) THEN
        NALARM% = NALARM% + 1
        ALARMNUM%(NALARM%) = I%
      END IF
    NEXT I%
    FOR I% = 1 TO NUM.FEATURES%
      IF FEATURE(I%) > DMEAN(E%, I%) + OP.NSTDEV(I%) * DSTDEV(E%, I%) THEN
        NALARM% = NALARM% + 1
        ALARMNUM%(NALARM%) = I% + NUM.FEATURES%
      END IF
    NEXT I%
    IF NALARM% > 0 THEN
      ALARM$ = YES$
      OPERATE.ALERT%(6) = 1
    END IF
```

```
    NF% = NUM.FEATURES%
    COLOR 1: LOCATE 16, 1
    PRINT "                DELX    DELY DIF.AN MID.AN  ANGL1  ANGL2  ANGR1  ANGR2";
    PRINT "  DELYL  DELYR"
    PRINT "  MEAN  ";
    FOR I% = 1 TO NF%: PRINT USING "#####.#"; DMEAN(E%, I%); : NEXT I%
    PRINT : PRINT " ST.DEV ";
    FOR I% = 1 TO NF%: PRINT USING "#####.#"; DSTDEV(E%, I%); : NEXT I%
    PRINT : PRINT "FEATURES";
    FOR I% = 1 TO NF%
      FOR J% = 1 TO NALARM%
        IF ALARMNUM%(J%) = I% OR ALARMNUM%(J%) = I% + NUM.FEATURES% THEN
          COLOR 3: J% = NALARM% + 1
        ELSE
          COLOR 1
        END IF
      NEXT J%
      PRINT USING "#####.#"; FEATURE(I%);
    NEXT I%
    COLOR 1
    PRINT
    RETURN

  RELAY.WALK: ' - routine to indicate to walk through the PMD.
    PORTVAL% = INP(PDISO%) OR PDISO.GO%
    OUT PDISO%, PORTVAL%
    CALL BLANKLINE(23): PRINT "Okay for "; DACTFIRST$(E%); " "; DACTINIT$(E%);
    PRINT " "; DACTLAST$(E%); " to walk through.";
    SOUND 100, 3: SOUND 50, 2
    RETURN

  RANDOM.ALARM: ' - routine to select an employee at random.
    IF RND(1) * 100 < PERCENT.ALARM% THEN
      ALARM$ = YES$
      OPERATE.ALERT%(7) = 1
    END IF
    RETURN

  ALARM.OFF: ' - routine to turn the alarm off
    PORTVAL% = INP(PDISO%) AND (255 - PDISO.ALARM%)
    OUT PDISO%, PORTVAL%
    TIMER OFF
    RETURN

  CLEAN.OFF: ' - routine to turn the CLEAN light off
    PORTVAL% = INP(PDISO%) AND (255 - PDISO.CLEAN%)
    OUT PDISO%, PORTVAL%
    TIMER OFF
    RETURN

  CHECK.KEYBOARD: ' - routine to check if key 'F1' has been hit
    A$ = INKEY$: LASTKEY$ = A$
```

```
    IF A$ = CHR$(123) THEN
      ATIME = TIMER: A$ = "": F1HIT$ = NO$
      WHILE TIMER - ATIME < 2 AND A$ = "" AND F1HIT$ = NO$
        A$ = INKEY$
        IF A$ = CHR$(200) THEN
          A$ = ""
          WHILE TIMER - ATIME < 2 AND A$ = "" AND F1HIT$ = NO$
            A$ = INKEY$
            IF A$ = CHR$(0) + CHR$(59) THEN F1HIT$ = YES$
          WEND
        END IF
      WEND
    END IF
    RETURN

  CHECK.INVALID.ENTRY: ' - routine to check for excessive signal sources.
    CALL GET.DATA
    IF IHT% > 0 THEN
      IF ABS(XAR%(COUNT%) - XAR%(IHT%)) > X.DEL.CHK.INV.S% THEN EVENT% = 1
      IF ABS(YAR%(COUNT%) - YAR%(IHT%)) > Y.DEL.CHK.INV.S% THEN EVENT% = 1
    END IF
    IF IRT% > 0 THEN
      IF ABS(XAR%(COUNT%) - XAR%(IRT%)) > X.DEL.CHK.INV.S% THEN EVENT% = 1
      IF ABS(YAR%(COUNT%) - YAR%(IRT%)) > Y.DEL.CHK.INV.S% THEN EVENT% = 1
    END IF
    IF EVENT% = 1 THEN
      EVENT% = 0
      IF ALARM.ON$ = NO$ THEN
        OUT PDISO%, PDISO.WAIT% + PDISO.ALARM%
        ALARM.ON$ = YES$
      END IF
    ELSE
      IF ALARM.ON$ = YES$ THEN
        GOSUB ALARM.OFF
        ALARM.ON$ = NO$
      END IF
    END IF
    RETURN

  RELAY.ALARM: ' - routine to set alarm and to display the alarm conditions.
  ' - turn on the relay to activate the 'ALARM' light.
    OUT PDISO%, PDISO.WAIT% + PDISO.ALARM%
    ON TIMER(ALARM.TIME%) GOSUB ALARM.OFF
    TIMER ON
    FOR I% = 1 TO NUM.OPERATE.ALERTS%
      IF OPERATE.ALERT%(I%) = 1 THEN ALERTNUM% = I%
    NEXT I%
    LPRINT MONTH$; " "; MID$(DATE$, 4, 2); " at "; TIME$; " ";
    IF ALERTNUM% = 1 THEN
      LPRINT "Employee "; ANAME$;
      SELECT CASE ALERTNUM%
      CASE 2
```

```
      LPRINT " is not calibrated."
    CASE 4
      LPRINT " took too long to enter."
    CASE 5
      LPRINT " took too long to walk through."
    CASE 6
      LPRINT " -features";
      FOR J% = 1 TO NALARM%: LPRINT " "; LTRIM$(STR$(ALARMNUM%(J%))); : NEXT J%
      LPRINT " detected."
    CASE 7
      LPRINT " selected randomly."
    CASE ELSE
      LPRINT "  alarm for unknown reason."
    END SELECT
  END IF
  RETURN


SET.BOARD: ' - routine to set up the frequency, phase, and gain.
  CALL TCINITIALIZATION
  NEWFREQ% = TCFREQSET%(FREQ&, PHASE%)
  NEWGAIN% = TCGAINSET%(GAIN%)
  CALL COMPENSATE
  RETURN

SAVE.DATABASE: ' - routine to save the database
  OPEN DATA.FILENAME$ FOR OUTPUT AS #1
  IF DRIVE.ERROR$ = NO$ THEN
  WRITE #1, NUMBER.OF.EMPLOYEES%
  IF NUMBER.OF.EMPLOYEES% > 0 THEN
    FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
      A1$ = DACTLAST$(I%)
      A2$ = DACTFIRST$(I%): A3$ = DACTINIT$(I%): A4% = DEMPNO%(I%)
      A5$ = DSEX(I%)
      A6 = DHEIGHT(I%): A7 = DWEIGHT(I%): A8$ = DBRIDGE(I%)
      A9$ = DIMPLANT(I%): AA$ = DPACE(I%): AB$ = DOTHER$(I%)
WRITE #1, A1$, A2$, A3$, A4%, A5$, A6, A6$, A7, A8$, A9$, AA$, AB$
      B1 = DMEAN(I%, 1): B2 = DMEAN(I%, 2): B3 = DMEAN(I%, 3)
      B4 = DMEAN(I%, 4): B5 = DMEAN(I%, 5): B6 = DMEAN(I%, 6)
      B7 = DMEAN(I%, 7): B8 = DMEAN(I%, 8): B9 = DMEAN(I%, 9)
      BA = DMEAN(I%, 10)
      WRITE #1, B1, B2, B3, B4, B5, B6, B7, B8, B9, BA
      B1 = DSTDEV(I%, 1): B2 = DSTDEV(I%, 2): B3 = DSTDEV(I%, 3)
      B4 = DSTDEV(I%, 4): B5 = DSTDEV(I%, 5): B6 = DSTDEV(I%, 6)
      B7 = DSTDEV(I%, 7): B8 = DSTDEV(I%, 8): B9 = DSTDEV(I%, 9)
      BA = DSTDEV(I%, 10)
      WRITE #1, B1, B2, B3, B4, B5, B6, B7, B8, B9, BA
      NEXT I%
    END IF
  DATA.CHANGE$ = NO$
  CLOSE #1
  END IF
```

```
    RETURN

EMPLOYEE.DATABASE: ' - routine to read the employee database
    A = FRE("A"): B = FRE(0): C = FRE(1): D = FRE(-1)
    DATA.FILENAME$ = "C:\PMD\DATA\EMPLOYE2.DAT"
    ON ERROR GOTO EMPLOYEE.DRIVE
    OPEN DATA.FILENAME$ FOR INPUT AS #1
    ON ERROR GOTO 0
    INPUT #1, NUMBER.OF.EMPLOYEES%
    IF NUMBER.OF.EMPLOYEES% > 0 THEN
      FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
        INPUT #1, DACTLAST$(I%), DACTFIRST$(I%), DACTINIT$(I%)
        INPUT #1, DEMPNO%(I%)
        INPUT #1, DSEX(I%), DHEIGHT(I%), DWEIGHT(I%), DBRIDGE(I%)
        INPUT #1, DIMPLANT(I%), DPACE(I%), DOTHER$(I%)
        FOR K% = 1 TO NUM.FEATURES%: INPUT #1, DMEAN(I%, K%): NEXT K%
        FOR K% = 1 TO NUM.FEATURES%: INPUT #1, DSTDEV(I%, K%): NEXT K%
        NEXT I%
      END IF
    CLOSE #1
    RETURN

EMPLOYEE.DRIVE: ' - routine to get the proper drive to read the
                  employee database.
    LOCATE 24, 10: PRINT "Cannot locate the data file "; DATA.FILENAME$
    ANS$ = "Y"
    WHILE ANS$ = "Y"
      LOCATE , 2: INPUT "Do you wish to enter a DOS command? (Y/N) "; ANS$
      IF ANS$ = "Y" THEN
        LOCATE , 10: INPUT "Enter the DOS command "; DOSCOMMAND$
        SHELL DOSCOMMAND$
        LOCATE , 1: INPUT "Press any key to continue"; A$
      END IF
    WEND
    LOCATE 24, 1: INPUT "Enter the new filename "; DATA.FILENAME$
    RESUME

CONSTANT.HEAD.DISPLAY: ' - routine to display the CONSTANT screen
    SCREEN 0: CLS
    FOR I% = 1 TO NUM.CON.HEAD%
      LOCATE CONHEADR%(I%), CONHEADC%(I%)
      PRINT CONHEADT$(I%);
      NEXT I%
    COLOR 10
    FOR I% = 1 TO NUM.CON.PARM%
      LOCATE CONROW%(I%), CONCOL%(I%): PRINT LEFT$(SPACECHR$, CONSPCS%(I%));
      LOCATE CONROW%(I%), CONCOL%(I%): PRINT CONPARM$(I%);
      NEXT I%
    COLOR 3
    RETURN

CONSTANT.DISPLAY.CHANGE: ' - routine to change the entries on the constant
```

```
                                    display.
  IPRM% = 1: IPRMMAX% = NUM.CON.PARM%
  FOR I% = 1 TO IPRMMAX%
    DATAR%(I%) = CONROW%(I%): DATAC%(I%) = CONCOL%(I%): SPCS%(I%) = CONSPCS%(I%)
    NEXT I%
  ALLDATA$ = "NOT OK"
  WHILE ALLDATA$ = "NOT OK"
    FINISH$ = NO$
    WHILE FINISH$ = NO$
      COLOR 10
      IF IPRM% > IPRMMAX% THEN IPRM% = 1
      IF IPRM% <= 0 THEN IPRM% = IPRMMAX%
      SOUND 100, 3: SOUND 50, 2
      ROW% = DATAR%(IPRM%): COLUMN% = DATAC%(IPRM%)
      LOCATE ROW%, COLUMN%
      CALL ADDNEWCHAR
      IF EXIT$ = YES$ THEN
        CALL BLANKLINE(25): SOUND 100, 3: SOUND 50, 2
        PRINT ; "Is it okay to quit without saving the changes? (Y/N) ";
        CALL GETINKEY: PRINT A$;
        IF A$ = "Y" OR A$ = "y" THEN RETURN
        CALL BLANKLINE(25)
      END IF
    WEND
    COLOR 3
    GOSUB CONSTANT.SETUP.NAME
    ALLDATA$ = "OK"
    WEND
  RETURN

CONSTANT.SETUP.NAME: ' - routine to name the constants
  FOR I% = 1 TO NUM.CON.PARM%
    B$ = " "
    FOR J% = 1 TO CONSPCS%(I%)
      A$ = CHR$(SCREEN(CONROW%(I%), CONCOL%(I%) + J% - 1))
      IF A$ = ASPACECHR$ THEN
        J% = CONSPCS%(I%) + 1
      ELSE
        B$ = B$ + A$
      END IF
    NEXT J%
    CONPARM$(I%) = RTRIM$(LTRIM$(B$))
  NEXT I%
  FREQ& = VAL(CONPARM$(1)): GAIN% = VAL(CONPARM$(2))
  PHASE% = VAL(CONPARM$(3)): DRANGE% = VAL(CONPARM$(4))
  LBND% = -DRANGE%: UBND% = DRANGE%
  X.DEL.CHK.S% = VAL(CONPARM$(5))
  X.DEL.CHK.E% = VAL(CONPARM$(6))
  RECOMP.CHECK% = 2 * X.DEL.CHK.S%
  Y.DEL.CHK.S% = X.DEL.CHK.S% / 2: Y.DEL.CHK.E% = X.DEL.CHK.E%
  CALIBRATE.MAX.DELAY.TIME% = VAL(CONPARM$(7))
  OPERATE.MAX.DELAY.TIME% = VAL(CONPARM$(8))
```

```
   CLEAN.TIME% = VAL(CONPARM$(17))
   ALARM.TIME% = VAL(CONPARM$(18))
   PERCENT.ALARM% = VAL(CONPARM$(19))
   PRINT.DISPLAY$ = CONPARM$(20)
   VIEW.LENGTH = VAL(CONPARM$(21))
   RETURN

ERROR.FOR.READ.NAME: ' - error routine for reading names and employee #
   ERRNUM% = ERR
   REASON$ = ""
   SELECT CASE ERRNUM%
   CASE 53
      REASON$ = "Cannot find file " + NAME.FILE$
   CASE 62
      REASON$ = "Tried to read past the end of the file."
   CASE 71
      REASON$ = "Disk is not ready - please ensure disc drive is ready."
   CASE ELSE
      REASON$ = "Error number " + STR$(ERRNUM%)
      REASON$ = REASON$ + " detected while accessing " + NAME.FILE$
   END SELECT
   RESUME NEXT

FEATURE.ERROR: ' - routine to check on the error occuring in FEATURE.
  FOR I% = 1 TO NUM.FEATURES%: FEATURE(I%) = 0: NEXT I%
  PASS.ALERT% = 1
  IF RUN.MODE$ = "OPERATE" THEN
    OPERATE.ALERT%(8) = 1
    ALARM$ = YES$
  END IF
  CALL GETFEATURES(1)
  RESUME NEXT

SUB ADDNEWCHAR
'  routine to add a character
   A$ = ""
   EXIT$ = "NO"
   WHILE A$ <> CHR$(13)
      CALL GETINKEY
      SELECT CASE ASC(A$)
         CASE 13: ' - to advance to the next field
            IPRM% = IPRM% + 1
            IF IPRM% > IPRMMAX% THEN IPRM% = 1
         CASE 32 TO 217: ' - standard characters
            PRINT A$; : A$ = "": C% = POS(0)
            IF (C% - COLUMN%) >= SPCS%(IPRM%) OR C% < COLUMN% THEN
               IPRM% = IPRM% + 1: A$ = CHR$(13)
               IF IPRM% > IPRMMAX% THEN IPRM% = 1
               END IF
         CASE 8: ' - backspace and overwrite
            LOCATE R%, C%: PRINT ASPACECHR$
            LOCATE R%, C% - 1: A$ = "": C% = POS(0)
```

```
            IF C% < COLUMN% THEN LOCATE R%, COLUMN%
         CASE 9: ' - tab to the next field
            IPRM% = IPRM% + 1
            IF IPRM% > IPRMMAX% THEN IPRM% = 1
            A$ = CHR$(13)
         CASE 27: ' - escape key - used to exit data entry - no checking
            A$ = CHR$(13)
            EXIT$ = "YES"
         CASE 0: ' - look at keys having 2 byte responses
            SELECT CASE ASC(RIGHT$(A$, 1))
               CASE 15: ' - shift tab - go back a field
                  IPRM% = IPRM% - 1
                  IF IPRM% <= 0 THEN IPRM% = IPRMMAX%
                  A$ = CHR$(13)
               CASE 75: ' - shift left 1 character
                  LOCATE R%, C% - 1
                  A$ = "": C% = POS(0)
                  IF C% < COLUMN% THEN LOCATE R%, COLUMN%
               CASE 77: ' - shift right 1 character
                  CP% = C% + 1
                  IF CP% <= 80 THEN LOCATE R%, CP%: A$ = "": C% = POS(0)
                  IF (C% - COLUMN%) >= SPCS%(IPRM%) OR C% = 80 THEN
                     IPRM% = IPRM% + 1: A$ = CHR$(13)
                     IF IPRM% > IPRMMAX% THEN IPRM% = 1
                     END IF
               CASE 68: ' - F10 - exit menu
                  FINISH$ = "YES"
                  A$ = CHR$(13)
               CASE ELSE
                  A$ = ""
               END SELECT
         CASE ELSE
         END SELECT
      WEND
END SUB

SUB BLANKLINE (LINE.TO.BLANK%)
' - routine to blank line 'LINE.TO.BLANK'
   LOCATE LINE.TO.BLANK%, 1
   PRINT SPACE$(80);
   LOCATE LINE.TO.BLANK%, 1
   EXIT SUB
END SUB

SUB CHECK.ID
' - This subprogram checks to ensure the EMPNO% is in the
'   the database and that the employee is calibrated.
'   Alarm conditions are set accordingly.

  PERSON.CALIBRATED$ = "NO": ALARM$ = "NO": PERSON.IN.DATABASE$ = "YES"

  IF RUN.MODE$ = "OPERATE" THEN
```

```
'    In the 'OPERATE' mode,  PERSON.WAITING$ ="YES" and it is necessary to
'    determine if a valid person is waiting based on whether the person
'    is in the database, and whether the person has been calibrated.
   INDEX.NUM% = 0
   FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
     IF EMPNO% = DEMPNO%(I%) THEN
       INDEX.NUM% = I%: I% = NUMBER.OF.EMPLOYEES% + 1
     END IF
   NEXT I%
   IF INDEX.NUM% = 0 THEN
     PERSON.IN.DATABASE$ = "NO"
     ALARM$ = "YES"
     OPERATE.ALERT%(1) = 1: ' - case for person not in the database.
     LPRINT "Person with employee number "; EMPNO%;
     LPRINT " is not in the database."
   END IF

'  - now need to make sure the employee has been calibrated
   IF PERSON.IN.DATABASE$ = "YES" THEN
     I% = INDEX.NUM%
     IF DMEAN(I%, 1) <> 0 THEN
       PERSON.CALIBRATED$ = "YES"
     ELSE
       ALARM$ = "YES": PERSON.CALIBRATED$ = "NO"
       OPERATE.ALERT%(2) = 1: ' - case for person not calibrated.
       LPRINT "Employee "; DACTFIRST$(I%); " "; DACTINIT$(I%); " ";
       LPRINT DACTLAST$(I%); " is not calibrated."
     END IF
   END IF
 END IF

' - Now test for the CALIBRATE mode
 IF RUN.MODE$ = "CALIBRATE" THEN
   INDEX.NUM% = 0
   FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
     IF DEMPNO%(I%) = EMPNO% THEN
       INDEX.NUM% = I%: I% = NUMBER.OF.EMPLOYEES% + 1
     END IF
   NEXT I%
   IF INDEX.NUM% = 0 THEN
     PERSON.IN.DATABASE$ = "NO": PERSON.CALIBRATED$ = "NO"
   ELSE
     IF DMEAN(INDEX.NUM%, 1) = 0 THEN PERSON.CALIBRATED$ = "NO"
   END IF
 END IF
 EXIT SUB

END SUB

SUB COMPENSATE
' - Routine to balance the signal
  COMPCHECK% = TCCOMPENSATION%(SEG COMP#(0))
```

```
    FOR J% = 1 TO 4: FOR I% = 1 TO TIMELOOP%: NEXT I%: NEXT J%
    CALL GET.DATA
    XREF% = XAVG%: YREF% = YAVG%
    X.AT.COMP% = XAVG%: Y.AT.COMP% = YAVG%
    IF COMPCHECK% <> 0 THEN
       LOCATE 21, 1: PRINT "Compensation not possible COMPCHECK%="; COMPCHECK%
       FOR I = 1 TO 3: PRINT COMP$(I); : NEXT I
       BEEP
       INPUT ; "   Press return to continue ", A$
    END IF
    EXIT SUB
END SUB

SUB COPY.EMPLOYEE.DATA (OLD%, NEW%)
'  this routine copies data from OLD% to NEW%
  DEMPNO%(NEW%) = DEMPNO%(OLD%)
  DACTLAST$(NEW%) = DACTLAST$(OLD%)
  DACTFIRST$(NEW%) = DACTFIRST$(OLD%)
  DACTINIT$(NEW%) = DACTINIT$(OLD%)
  DSEX(NEW%) = DSEX(OLD%): DHEIGHT(NEW%) = DHEIGHT(OLD%)
  DWEIGHT(NEW%) = DWEIGHT(OLD%): DBRIDGE(NEW%) = DBRIDGE(OLD%)
  DIMPLANT(NEW%) = DIMPLANT(OLD%): DPACE(NEW%) = DPACE(OLD%)
  DOTHER$(NEW%) = DOTHER$(OLD%)
  FOR I% = 1 TO NUM.FEATURES%
    DMEAN(NEW%, I%) = DMEAN(OLD%, I%)
    DSTDEV(NEW%, I%) = DSTDEV(OLD%, I%)
  NEXT I%
  EXIT SUB
END SUB

SUB DELETE.EMPLOYEE.FROM.DATABASE
' - routine to subtract an employee
  J% = INDEX.NUM% + 1
  WHILE J% <= NUMBER.OF.EMPLOYEES%
    JM% = J% - 1
    DEMPNO%(JM%) = DEMPNO%(J%)
    DACTLAST$(JM%) = DACTLAST$(J%)
    DACTFIRST$(JM%) = DACTFIRST$(J%): DACTINIT$(JM%) = DACTINIT$(J%)
    DSEX(JM%) = DSEX(J%): DHEIGHT(JM%) = DHEIGHT(J%)
    DWEIGHT(JM%) = DWEIGHT(J%): DBRIDGE(JM%) = DBRIDGE(J%)
    DIMPLANT(JM%) = DIMPLANT(J%): DPACE(JM%) = DPACE(J%)
    DOTHER$(JM%) = DOTHER$(J%)
    FOR K% = 1 TO NUM.FEATURES%
      DMEAN(JM%, K%) = DMEAN(J%, K%)
      DSTDEV(JM%, K%) = DSTDEV(J%, K%)
    NEXT K%
    J% = J% + 1
  WEND
  J% = NUMBER.OF.EMPLOYEES%
  DEMPNO%(J%) = 0:  DACTLAST$(J%) = ""
  DACTFIRST$(J%) = "": DACTINIT$(J%) = ""
  DSEX(J%) = "": DHEIGHT(J%) = 0: DWEIGHT(J%) = 0: DBRIDGE(J%) = ""
```

```
    DIMPLANT(J%) = "": DPACE(J%) = "": DOTHER$(J%) = ""
    FOR K% = 1 TO NUM.FEATURES%
      DMEAN(J%, K%) = 0
      DSTDEV(J%, K%) = 0
    NEXT K%

    NUMBER.OF.EMPLOYEES% = NUMBER.OF.EMPLOYEES% - 1
    DATA.CHANGE$ = "YES"
    EXIT SUB

END SUB

SUB DISPLAY.DATABASE
' - routine to display the database
  EMAX% = NUMBER.OF.EMPLOYEES%
  REDIM SEQ.NUM%(EMAX%), SORTED%(EMAX%)
  SCREEN 0: CLS : COLOR 10
  LOCATE 1, 20: PRINT CHR$(201) + STRING$(21, CHR$(205)) + CHR$(187)
  LOCATE 2, 20: PRINT CHR$(186) + "   DATABASE VALUES   " + CHR$(186)
  LOCATE 3, 20: PRINT CHR$(200) + STRING$(21, CHR$(205)) + CHR$(188)
  CALL BLANKLINE(25): COLOR 10, 1: PRINT "Press 'Esc' key to exit.";
  COLOR 2: LOCATE 5, 2
  PRINT "Do you want a print-out (P) or a display (D) of the database ? ";
  CALL GETINKEY: PRINT A$
  IF A$ = CHR$(27) THEN VIEW PRINT: CLS : EXIT SUB
  GOSUB PRINT.RESEQUENCE
  IF A$ = "P" OR A$ = "p" THEN
    DBHEAD$ = "PRECIOUS METAL DETECTOR - DATABASE"
    DATE.TIME$ = "Date : " + DATE$ + "   Time : " + TIME$
    NUMBER.PER.PAGE% = 50
    IBOT% = 0: PAGENUM% = 0: I% = 1
     LPRINT CHR$(27) + "n"; : ' - sets printer to NLQ pica pitch
    PRINT "Positon the page in the printer to the top of the page."
    INPUT ; "      Press return to continue. ", A$
    LPRINT CHR$(27) + "@": ' - initialize the printer
    WHILE I% <= NUMBER.OF.EMPLOYEES%
      PAGENUM% = PAGENUM% + 1
      ITOP% = IBOT% + 1: IBOT% = IBOT% + NUMBER.PER.PAGE%
      IF PAGENUM% > 1 THEN LPRINT CHR$(12): ' -    form feed
      LPRINT "    "; DATE.TIME$; SPACE$(65 - LEN(DATE.TIME$));
      LPRINT "Page "; PAGENUM%: LPRINT
      LPRINT SPACE$(40 - LEN(DBHEAD$) / 2); DBHEAD$
      LPRINT SPACE$(40 - LEN(DBHEAD$) / 2 - 1);
      LPRINT STRING$(LEN(DBHEAD$) + 2, "-")
      LPRINT " EMPLOYEE"
      LPRINT "  NUMBER  LAST NAME       FIRST NAME      INITIAL    ";
      LPRINT "CALIBRATED"
      LPRINT "  ------  ----------      ----------      -------    ";
      LPRINT "----------"
      WHILE I% <= NUMBER.OF.EMPLOYEES% AND I% <= IBOT%
        J% = SEQ.NUM%(I%)
        LPRINT USING "########"; DEMPNO%(J%);
```

```
        LPRINT "  ";
        LPRINT USING "\                \"; DACTLAST$(J%);
        LPRINT USING "\                \"; DACTFIRST$(J%);
        LPRINT USING "\ \ "; DACTINIT$(J%);
        LPRINT "      ";
        IF DMEAN(J%, 1) <> 0 THEN
          LPRINT "YES";
        ELSE
          LPRINT " NO";
        END IF
        IF LEN(DACTINIT$(J%)) > 1 THEN
          IF RIGHT$(DACTINIT$(J%), 1) = "V" THEN LPRINT "    V"
          IF RIGHT$(DACTINIT$(J%), 1) = "T" THEN LPRINT "    T"
        ELSE
          LPRINT
        END IF
        I% = I% + 1
      WEND
    WEND
    'LPRINT CHR$(27) + "P"; :' - reset to draft mode
    LPRINT CHR$(12); ' -  form feed
  ELSE
    LOCATE 5, 1: PRINT SPACE$(79); : LOCATE 4, 1
    PRINT " EMPLOYEE"
    PRINT "  "; : COLOR 1: PRINT "NUMBER"; : COLOR 2: PRINT "   ";
    COLOR 1: PRINT "LAST NAME"; : COLOR 2: PRINT "       ";
    COLOR 1: PRINT "FIRST NAME"; : COLOR 2: PRINT "      ";
    COLOR 1: PRINT "INITIAL"; : COLOR 2: PRINT "     ";
    COLOR 1: PRINT "CALIBRATED": COLOR 2
    VIEW PRINT 6 TO 24
    IEND% = 18
    I% = 1
    WHILE I% <= NUMBER.OF.EMPLOYEES%
      WHILE I% <= IEND% AND I% <= NUMBER.OF.EMPLOYEES%
        J% = SEQ.NUM%(I%)
        PRINT USING "########"; DEMPNO%(J%);
        PRINT "  ";
        PRINT USING "\                \"; DACTLAST$(J%);
        PRINT USING "\                \"; DACTFIRST$(J%);
        PRINT USING "\ \ "; LEFT$(DACTINIT$(J%), 1);
        PRINT "      ";
        IF DMEAN(J%, 1) <> 0 THEN
          PRINT "YES";
        ELSE
          PRINT " NO";
        END IF
        IF LEN(DACTINIT$(J%)) > 1 THEN
          IF RIGHT$(DACTINIT$(J%), 1) = "V" THEN PRINT "    V"
          IF RIGHT$(DACTINIT$(J%), 1) = "T" THEN PRINT "    T"
        ELSE
          PRINT
        END IF
```

```
        I% = I% + 1
      WEND
      AROW% = CSRLIN: ACOL% = POS(0)
      IEND% = I% + 18
      CALL BLANKLINE(25)
      INPUT ; "Press return to continue", A$
      LOCATE AROW%, ACOL%
    WEND
  END IF
  VIEW PRINT
  EXIT SUB

PRINT.RESEQUENCE: ' - routine to determine the sequence to print the data
'   in an alphabetical order.
  NSORT% = 0: LSORTPOS% = 0: HSORTPOS% = EMAX% + 1
  WHILE NSORT% < EMAX%
    FOR I% = 1 TO EMAX%
      IF SORTED%(I%) = 0 THEN
        MIN$ = DACTLAST$(I%) + DACTFIRST$(I%) + DACTINIT$(I%)
        MAX$ = MIN$
        SORTL% = I%: SORTH% = I%
        I% = EMAX% + 1
      END IF
    NEXT I%
    FOR I% = 1 TO EMAX%
      IF SORTED%(I%) = 0 THEN
        ASTR$ = DACTLAST$(I%) + DACTFIRST$(I%) + DACTINIT$(I%)
        IF ASTR$ < MIN$ THEN MIN$ = ASTR$: SORTL% = I%
        IF ASTR$ > MAX$ THEN MAX$ = ASTR$: SORTH% = I%
      END IF
    NEXT I%
    NSORT% = NSORT% + 1
    LSORTPOS% = LSORTPOS% + 1
    SEQ.NUM%(LSORTPOS%) = SORTL%
    SORTED%(SORTL%) = 1
    IF SORTL% <> SORTH% THEN
      NSORT% = NSORT% + 1
      HSORTPOS% = HSORTPOS% - 1
      SEQ.NUM%(HSORTPOS%) = SORTH%
      SORTED%(SORTH%) = 1
    END IF
  WEND
  RETURN

END SUB

SUB GET.DATA
' - routine to collect a sample
'   This routine is used when checking for an invalid entry, and when the
'   balancing of the card is required.
  CALL TCACQUISITION(SEG A%(0)): Z% = 1: SX& = A%(0): SY& = A%(1)
  OLDTIME = TIMER
```

```
    WHILE TIMER = OLDTIME
      CALL TCACQUISITION(SEG A%(0))
      SX& = SX& + A%(0): SY& = SY& + A%(1): Z% = Z% + 1
    WEND
    XAVG% = SX& / Z%: YAVG% = SY& / Z%
    OLDTIME = NEWTIME
    COUNT% = COUNT% + 1: IF COUNT% > MAX.SAMPLES.FOR.EVENT% THEN COUNT% = 0
    IRT% = IRT% + 1: IF IRT% > MAX.SAMPLES.FOR.EVENT% THEN IRT% = 0
    IHT% = IHT% + 1: IF IHT% > MAX.SAMPLES.FOR.EVENT% THEN IHT% = 0
    XAR%(COUNT%) = XAVG%
    YAR%(COUNT%) = YAVG%
    EXIT SUB
  END SUB

  SUB GET.DAYS.FROM.1990
  ' - this procedure gets the number of days since JAN 1,1990

    ADATE$ = DATE$
    YREND% = VAL(RIGHT$(ADATE$, 4))
    ANUM& = 0: YR% = 1990
    WHILE YR% < YREND%
      ANUM& = ANUM& + 365
      IF (YR% MOD 4) = 0 THEN ANUM% = ANUM% + 1
      YR% = YR% + 1
    WEND
    MONTHEND% = VAL(LEFT$(ADATE$, 2))
    MONTH% = 1
    WHILE MONTH% < MONTHEND%
      ANUM& = ANUM& + DM%(MONTH%)
      MONTH% = MONTH% + 1
    WEND
    IF (YREND% MOD 4) = 0 THEN
      IF MONTHEND% > 2 THEN ANUM& = ANUM& + 1
    END IF
    DAYS.FROM.1990% = ANUM& + VAL(MID$(ADATE$, 4, 2))
    EXIT SUB

  END SUB

  SUB GETFEATURES (condition%)
  ' - Procedure to determine the features in the detected event
  '  -Features include amplitude maximums, mean slopes,
  '   and midrange Y separations.

    IF condition% = 1 THEN GOTO END.FEATURE.ERROR

  ' - need to adjust the values to allow for some drift
    NAVG% = 4
  ' - get average value for the first point at the start of the signal
    SUMX = 0: SUMY = 0
    IST% = 1: IEND% = IST% + NAVG% - 1: IREFST% = IEND%
    FOR I% = IST% TO IEND%: SUMX = SUMX + X%(I%): SUMY = SUMY + Y%(I%): NEXT I%
```

```
  AVGXST = SUMX / NAVG%: AVGYST = SUMY / NAVG%
' - get average value for the last point at the end of the signal
  SUMX = 0: SUMY = 0
  IST% = NSAMP% - NAVG% + 1: IEND% = NSAMP%: IREFEND% = IST%
  FOR I% = IST% TO IEND%: SUMX = SUMX + X%(I%): SUMY = SUMY + Y%(I%): NEXT I%
  AVGXEND = SUMX / NAVG%: AVGYEND = SUMY / NAVG%
  DELXPERPOINT = (AVGXEND - AVGXST) / (IREFEND% - IREFST%)
  DELYPERPOINT = (AVGYEND - AVGYST) / (IREFEND% - IREFST%)
' - adjust values to make end points match
  J% = 1
  FOR I% = IREFST% + 1 TO IREFEND% - 1
    J% = J% + 1
    X%(J%) = X%(I%) - DELXPERPOINT * (J% - 1)
    Y%(J%) = Y%(I%) - DELYPERPOINT * (J% - 1)
  NEXT I%
  NSAMP% = J% + 1
  X%(1) = AVGXST: Y%(1) = AVGYST
  X%(NSAMP%) = X%(1): Y%(NSAMP%) = Y%(1)

' - now I need to find the maximum ranges and the points where they occur.
  XMIN% = X%(1): XMAX% = X%(1)
  YMIN% = Y%(1): YMAX% = Y%(1)
  CXMIN% = 0: CXMAX% = 0: CYMIN% = 0: CYMAX% = 0
  FOR I% = 1 TO NSAMP%
    IF X%(I%) < XMIN% THEN XMIN% = X%(I%): CXMIN% = I%
    IF X%(I%) > XMAX% THEN XMAX% = X%(I%): CXMAX% = I%
    IF Y%(I%) < YMIN% THEN YMIN% = Y%(I%): CYMIN% = I%
    IF Y%(I%) > YMAX% THEN YMAX% = Y%(I%): CYMAX% = I%
  NEXT I%
  IF CXMIN% > 2 THEN XMIN% = (X%(CXMIN% - 1) + XMIN% + X%(CXMIN% + 1)) / 3
  IF CXMAX% > 2 THEN XMAX% = (X%(CXMAX% - 1) + XMAX% + X%(CXMAX% + 1)) / 3
  IF CYMIN% > 2 THEN YMIN% = (Y%(CYMIN% - 1) + YMIN% + Y%(CYMIN% + 1)) / 3
  IF CYMAX% > 2 THEN YMAX% = (Y%(CYMAX% - 1) + YMAX% + Y%(CYMAX% + 1)) / 3
  DELXMAX% = XMAX% - XMIN%
  DELYMAX% = YMAX% - YMIN%
  FEATURE(1) = DELXMAX%: FEATURE(2) = DELYMAX%

' - now I need to get the slope between the points determined from the
    maximum change
  IF DELXMAX% > DELYMAX% THEN
    IF CXMIN% < CXMAX% THEN
      IST% = CXMIN%: IEND% = CXMAX%: GOSUB GET.SLOPE
      ANG = RADTODEG * ATN(SLOPE)
    ELSE
      IST% = CXMAX%: IEND% = CXMIN%: GOSUB GET.SLOPE
      ANG = RADTODEG * ATN(SLOPE) + 180
    END IF
  ELSE
    IF CYMIN% < CYMAX% THEN
      IST% = CYMIN%: IEND% = CYMAX%: GOSUB GET.SLOPE
      ANG = RADTODEG * ATN(SLOPE)
      IF SLOPE < 0 THEN ANG = ANG + 180
```

```
    ELSE
      IST% = CYMAX%: IEND% = CYMIN%: GOSUB GET.SLOPE
      ANG = RADTODEG * ATN(SLOPE)
      IF SLOPE > 0 THEN ANG = ANG - 180
    END IF
  END IF
  IF SLOPE = 9999 THEN ANG = 999
  FEATURE(3) = ANG

' - now I will get the slope between the middle of the left wing and the
'   middle of the right wing obtained as the signal goes from left to right.
  XLOC.LEFT% = (XMIN% + X%(1)) / 2
  XLOC.RIGHT% = (XMAX% + X%(NSAMP%)) / 2
  I% = CXMIN%: CHECKX% = XLOC.LEFT%
  WHILE I% <= NSAMP% AND X%(I%) < CHECKX%: I% = I% + 1: WEND: IST% = I% - 1
  I% = CXMAX%: CHECKX% = XLOC.RIGHT%
  WHILE I% >= 0 AND X%(I%) > CHECKX%: I% = I% - 1: WEND: IEND% = I% + 1
  GOSUB GET.SLOPE
  IF SLOPE = 9999 THEN
    ANG = 999
  ELSE
    ANG = RADTODEG * ATN(SLOPE)
  END IF
  FEATURE(4) = ANG

' - now I will get the slope and mean Y values at the midpoints between
'   the balance points and the left and right extremes of the X-signal
' - first to get the values on the left of the balance point
  DX.LEFT% = ABS((X%(1) - XMIN%)) / 5
  I% = CXMIN%: CHECKX% = XLOC.LEFT% - DX.LEFT%
  WHILE I% >= 0 AND X%(I%) < CHECKX%: I% = I% - 1: WEND: IEND% = I% + 1
  I% = CXMIN%: CHECKX% = XLOC.LEFT% + DX.LEFT%
  WHILE I% >= 0 AND X%(I%) < CHECKX%: I% = I% - 1: WEND: IST% = I%
  GOSUB GET.SLOPE
  IF SLOPE = 9999 THEN
    ANG.LEFT1 = 999: YMEAN.LEFT1 = 999
  ELSE
    YMEAN.LEFT1 = SUMY / NP: ANG.LEFT1 = RADTODEG * ATN(SLOPE) + 180
  END IF

  I% = CXMIN%: CHECKX% = XLOC.LEFT% - DX.LEFT%
  WHILE I% <= NSAMP% AND X%(I%) < CHECKX%: I% = I% + 1: WEND: IST% = I% - 1
  I% = CXMIN%: CHECKX% = XLOC.LEFT% + DX.LEFT%
  WHILE I% <= NSAMP% AND X%(I%) < CHECKX%: I% = I% + 1: WEND: IEND% = I%
  GOSUB GET.SLOPE
  IF SLOPE = 9999 THEN
    ANG.LEFT2 = 999: YMEAN.LEFT2 = 999
  ELSE
    YMEAN.LEFT2 = SUMY / NP: ANG.LEFT2 = RADTODEG * ATN(SLOPE)
  END IF

' - now to get the values on the right of the balance point
```

```
  DX.RIGHT% = (XMAX% - X%(NSAMP%)) / 5
  I% = CXMAX%: CHECKX% = XLOC.RIGHT% + DX.RIGHT%
  WHILE I% >= 0 AND X%(I%) > CHECKX%: I% = I% - 1: WEND: IEND% = I% + 1
  I% = CXMAX%: CHECKX% = XLOC.RIGHT% - DX.RIGHT%
  WHILE I% >= 0 AND X%(I%) > CHECKX%: I% = I% - 1: WEND: IST% = I%
  GOSUB GET.SLOPE
  IF SLOPE = 9999 THEN
    ANG.RIGHT1 = 999: YMEAN.RIGHT1 = 999
  ELSE
    YMEAN.RIGHT1 = SUMY / NP: ANG.RIGHT1 = RADTODEG * ATN(SLOPE)
  END IF

  I% = CXMAX%: CHECKX% = XLOC.RIGHT% + DX.RIGHT%
  WHILE I% <= NSAMP% AND X%(I%) > CHECKX%: I% = I% + 1: WEND: IST% = I% - 1
  I% = CXMAX%: CHECKX% = XLOC.RIGHT% - DX.RIGHT%
  WHILE I% <= NSAMP% AND X%(I%) > CHECKX%: I% = I% + 1: WEND: IEND% = I%
  GOSUB GET.SLOPE
  IF SLOPE = 9999 THEN
    ANG.RIGHT2 = 999: YMEAN.RIGHT2 = 999
  ELSE
    YMEAN.RIGHT2 = SUMY / NP: ANG.RIGHT2 = RADTODEG * ATN(SLOPE) + 180
  END IF

  FEATURE(5) = ANG.LEFT1: FEATURE(6) = ANG.LEFT2
  FEATURE(7) = ANG.RIGHT1: FEATURE(8) = ANG.RIGHT2
  FEATURE(9) = YMEAN.LEFT1 - YMEAN.LEFT2
  FEATURE(10) = YMEAN.RIGHT1 - YMEAN.RIGHT2

END.FEATURE.ERROR: ' - where FEATURE.ERROR resumes
  ON ERROR GOTO 0
  EXIT SUB


GET.SLOPE: ' - routine to get the mean slope between points IST% and IEND%
'            - a linear regression formulation is used.
  SUMX = 0: SUMY = 0: SUMXY = 0: SUMXSQ = 0
  IF IST% < 0 THEN IST% = 0
  IF IEND% < 0 THEN IEND% = 0
  NP = IEND% - IST% + 1
  ASTEP% = 1: IF IEND% < IST% THEN ASTEP% = -1
  FOR I% = IST% TO IEND% STEP ASTEP%
    SUMX = SUMX + X%(I%)
    SUMY = SUMY + Y%(I%)
    SUMXY = SUMXY + 1! * X%(I%) * Y%(I%)
    SUMXSQ = SUMXSQ + 1! * X%(I%) * X%(I%)
  NEXT I%
  DIVISOR = NP * SUMXSQ - SUMX * SUMX
  NUMERATOR = NP * SUMXY - SUMY * SUMX
  IF ABS(DIVISOR) < 1 THEN
    SLOPE = 9999
  ELSE
    SLOPE = NUMERATOR / DIVISOR
```

```
  END IF
  RETURN

END SUB

SUB GETINKEY
' - routine to get a character using the INKEY$ command
   A$ = ""
   WHILE A$ = ""
      R% = CSRLIN: C% = POS(0): ACHAR = SCREEN(R%, C%): PRINT " ";
      IF ACHAR <= 32 THEN ACHAR = ASC(ASPACECHR$)
      LOCATE R%, C%
      FOR I% = 1 TO TIMELOOP%: NEXT I%: PRINT CHR$(ACHAR);
      FOR I% = 1 TO TIMELOOP%: NEXT I%: LOCATE R%, C%
      A$ = INKEY$
   WEND
END SUB

SUB GETSIGNATURE
' - routine to acquire the signature as the person walks
'                   through the metal detector.

' - initialize some parameters
  EVENT% = 0: COUNT% = -1: TOTAL.COUNT% = -1: OLDTIME = TIMER
  START.EVENT% = -1: END.EVENT% = -1
  IRT% = COUNT% - SAMPLES.IN.RISETIME%
  IHT% = COUNT% - SAMPLES.IN.RISETIME% / 2
  GOSUB SGET.DATA
  XT1% = XAR%(COUNT%): YT1% = YAR%(COUNT%)
  PSET (XT1% - XREF%, YT1% - YREF%)

'  checking for the event occuring before 1 full rise time
  WHILE COUNT% < SAMPLES.IN.RISETIME%
    GOSUB SGET.DATA
    LINE -(XAR%(COUNT%) - XREF%, YAR%(COUNT%) - YREF%)
    IF EVENT% = 0 THEN
      IF ABS(XAR%(COUNT%) - XT1%) > X.DEL.CHK.S% THEN
        EVENT% = 1: DETECT.COUNT% = COUNT%
      END IF
      IF ABS(YAR%(COUNT%) - YT1%) > Y.DEL.CHK.S% THEN
        EVENT% = 1: DETECT.COUNT% = COUNT%
      END IF
    ELSE
      IF RELAY.CONTROLS$ = YES$ THEN
        PORTVAL% = INP(PDISO%) AND (255 - PDISO.GO%)
        OUT PDISO%, PORTVAL%
      END IF
    END IF
  WEND
  IF EVENT% = 1 THEN
    START.EVENT% = 0
    WHILE TOTAL.COUNT% < SAMPLES.IN.RISETIME%
```

```
      GOSUB SGET.DATA
      LINE -(XAR%(COUNT%) - XREF%, YAR%(COUNT%) - YREF%)
    WEND
  END IF
  ' now need to check for an event after 1 risetime
  IF EVENT% = 0 THEN
    PROBLEM% = 0
    WHILE EVENT% = 0 AND PROBLEM% = 0
      GOSUB SGET.DATA
      LINE -(XAR%(COUNT%) - XREF%, YAR%(COUNT%) - YREF%)
      IF ABS(XAR%(COUNT%) - XAR%(IRT%)) > X.DEL.CHK.S% THEN EVENT% = 1
      IF ABS(XAR%(COUNT%) - XAR%(IHT%)) > X.DEL.CHK.S% THEN EVENT% = 1
      IF ABS(YAR%(COUNT%) - YAR%(IRT%)) > Y.DEL.CHK.S% THEN EVENT% = 1
      IF ABS(YAR%(COUNT%) - YAR%(IHT%)) > Y.DEL.CHK.S% THEN EVENT% = 1
      IF TOTAL.COUNT% > MAX.SAMPLES.BEFORE.EVENT% THEN
        PROBLEM% = 1
        PASS.ALERT% = 1
        IF RELAY.CONTROLS$ = YES$ THEN
          IF RUN.MODE$ = "OPERATE" THEN
            OPERATE.ALERT%(4) = 1: ALARM$ = YES$
          END IF
          IF RUN.MODE$ = "CALIBRATE" THEN
            OUT PDISO%, PDISO.WAIT%
          END IF
        END IF
      END IF
    WEND
    IF EVENT% = 1 THEN
      IF RELAY.CONTROLS$ = YES$ THEN
        PORTVAL% = INP(PDISO%) AND (255 - PDISO.GO%)
        OUT PDISO%, PORTVAL%
      END IF
      START.EVENT% = IRT%
    END IF
    IF PROBLEM% = 1 THEN
      LPRINT NAME$; " waited too long before entering the portal."
    END IF
  END IF

  ' - if an event has been detected, the end of the event must be determined
  IF EVENT% = 1 THEN
    PROBLEM% = 0
    SAMPLES.TO.ALERT% = TOTAL.COUNT% + MAX.SAMPLES.FOR.EVENT%
    WHILE EVENT% = 1 AND PROBLEM% = 0
      GOSUB SGET.DATA
      LINE -(XAR%(COUNT%) - XREF%, YAR%(COUNT%) - YREF%)
      IF ABS(XAR%(COUNT%) - XAR%(IRT%)) < X.DEL.CHK.E% THEN
        IF ABS(XAR%(COUNT%) - XAR%(IHT%)) < X.DEL.CHK.E% THEN
          IF ABS(YAR%(COUNT%) - YAR%(IRT%)) < Y.DEL.CHK.E% THEN
            IF ABS(YAR%(COUNT%) - YAR%(IHT%)) < Y.DEL.CHK.E% THEN
              EVENT% = 0
              END.EVENT% = COUNT%
```

```
            END IF
          END IF
        END IF
      END IF
      IF TOTAL.COUNT% > SAMPLES.TO.ALERT% THEN
        PROBLEM% = 1
        PASS.ALERT% = 1
        OPERATE.ALERT%(5) = 1: ALARM$ = YES$
      END IF
    WEND
    IF PROBLEM% = 1 THEN
      LPRINT NAME$; " took too long to walk through the portal."
    END IF
  END IF

' - need to obtain a new properly sequenced array.
  IF END.EVENT% > 0 THEN
    J% = 0
    IF END.EVENT% > START.EVENT% THEN
      FOR I% = START.EVENT% TO END.EVENT%
        J% = J% + 1: X%(J%) = XAR%(I%): Y%(J%) = YAR%(I%)
      NEXT I%
    ELSE
      FOR I% = START.EVENT% TO MAX.SAMPLES.FOR.EVENT%
        J% = J% + 1: X%(J%) = XAR%(I%): Y%(J%) = YAR%(I%)
      NEXT I%
      FOR I% = 0 TO END.EVENT%
        J% = J% + 1: X%(J%) = XAR%(I%): Y%(J%) = YAR%(I%)
      NEXT I%
    END IF
    NSAMP% = J%
  END IF
  EXIT SUB

SGET.DATA: ' - routine to collect a sample
' - This routine is called from GETSIGNATURE, and needs TOTAL.COUNT%
  CALL TCACQUISITION(SEG A%(0)): SX& = A%(0): SY& = A%(1): Z% = 1
  WHILE TIMER = OLDTIME
    CALL TCACQUISITION(SEG A%(0))
    SX& = SX& + A%(0): SY& = SY& + A%(1): Z% = Z% + 1
  WEND
  OLDTIME = TIMER
  XAVG% = SX& / Z%: YAVG% = SY& / Z%
  TOTAL.COUNT% = TOTAL.COUNT% + 1
  COUNT% = COUNT% + 1: IF COUNT% > MAX.SAMPLES.FOR.EVENT% THEN COUNT% = 0
  IRT% = IRT% + 1: IF IRT% > MAX.SAMPLES.FOR.EVENT% THEN IRT% = 0
  IHT% = IHT% + 1: IF IHT% > MAX.SAMPLES.FOR.EVENT% THEN IHT% = 0
  XAR%(COUNT%) = XAVG%
  YAR%(COUNT%) = YAVG%
  RETURN

END SUB
```

```
SUB MAIN.MENU.DISPLAY
' - procedure to display the main menu
  SCREEN 0, , 1, 1: CLS : COLOR 10
  VIEW PRINT
  LOCATE 1, 1: PRINT CHR$(201) + STRING$(78, 205) + CHR$(187);
  LOCATE 2, 1: PRINT CHR$(186) + SPACE$(78) + CHR$(186);
  LOCATE 3, 1: PRINT CHR$(200) + STRING$(78, 205) + CHR$(188);
  COLOR 9: LOCATE 2, 18: PRINT "THE WISE COMPANY'S  PRECIOUS METAL DETECTOR"
  COLOR 2
  LOCATE 7, 10: PRINT "Select - "; : COLOR 9: PRINT "C";
  COLOR 2: PRINT " - to calibrate an employee."
  LOCATE 9, 19: COLOR 9: PRINT "O";
  COLOR 2: PRINT " - to operate in the detection mode."
  LOCATE 11, 19: COLOR 9: PRINT "U";
  COLOR 2: PRINT " - to update the employee data base."
  LOCATE 13, 19: COLOR 9: PRINT "D";
  COLOR 2: PRINT " - to display the database."
  LOCATE 17, 2: PRINT "Enter your selection ( ";
  COLOR 9: PRINT "C, O, U, "; : COLOR 2: PRINT " or "; : COLOR 9: PRINT "D";
  COLOR 2: PRINT " ) ";
  EXIT SUB
END SUB

SUB PMDCONS (A%)

  GOSUB PROGRAM.MENU.CONSTANTS
  GOSUB PROGRAM.CONSTANTS
  EXIT SUB

PROGRAM.MENU.CONSTANTS: ' - routine used to set up the program constants
'                         used in the MENU.CONSTANTS routine.
  OPEN "C:\PMD\DATA\LASTSAVE.DAT" FOR INPUT AS #1
  INPUT #1, FILE.CONSTANTS$
  CLOSE #1
  OPEN FILE.CONSTANTS$ FOR INPUT AS #1
  INPUT #1, LAST.DATE$, LAST.TIME$
  INPUT #1, X.DEL.CHK.S%, Y.DEL.CHK.S%, X.DEL.CHK.E%, Y.DEL.CHK.E%
  INPUT #1, CALIBRATE.MAX.DELAY.TIME%, OPERATE.MAX.DELAY.TIME%
  INPUT #1, VIEW.LENGTH
  INPUT #1, FREQ&, PHASE%, GAIN%, DRANGE%
  INPUT #1, CLEAN.TIME%, ALARM.TIME%, PERCENT.ALARM%
  CLOSE #1
  LBND% = -DRANGE%: UBND% = DRANGE%: ' - sets display limits of digitized data.
  CONPARM$(1) = LTRIM$(STR$(FREQ&)): CONPARM$(2) = LTRIM$(STR$(GAIN%))
  CONPARM$(3) = LTRIM$(STR$(PHASE%)): CONPARM$(4) = LTRIM$(STR$(DRANGE%))
  CONPARM$(5) = LTRIM$(STR$(X.DEL.CHK.S%))
  CONPARM$(6) = LTRIM$(STR$(X.DEL.CHK.E%))
  CONPARM$(7) = LTRIM$(STR$(CALIBRATE.MAX.DELAY.TIME%))
  CONPARM$(8) = LTRIM$(STR$(OPERATE.MAX.DELAY.TIME%))
  CONPARM$(17) = LTRIM$(STR$(CLEAN.TIME%))
  CONPARM$(18) = LTRIM$(STR$(ALARM.TIME%))
  CONPARM$(19) = LTRIM$(STR$(PERCENT.ALARM%))
```

```
  CONPARM$(21) = LTRIM$(STR$(VIEW.LENGTH))
  RETURN

PROGRAM.CONSTANTS: ' - routine used to set up the other program constants
  M$(1) = "Jan": M$(2) = "Feb": M$(3) = "Mar": M$(4) = "Apr": M$(5) = "May"
  M$(6) = "Jun": M$(7) = "Jul": M$(8) = "Aug": M$(9) = "Sep": M$(10) = "Oct"
  M$(11) = "Nov": M$(12) = "Dec"
  DM%(1) = 31: DM%(2) = 28: DM%(3) = 31: DM%(4) = 30
  DM%(5) = 31: DM%(6) = 30: DM%(7) = 31: DM%(8) = 31
  DM%(9) = 30: DM%(10) = 31: DM%(11) = 30: DM%(12) = 31
  INTERRUPT.RATE = 18.2
  RISETIME = 1!
  DIFFERENTIAL.TIME = .75
  MAX.WALK.TIME% = 10
  REFLINE = 21
  PI = 4 * ATN(1): RADTODEG = 180 / PI
  PDISO% = &H2A0
  PDISO.WAIT% = 0: PDISO.GO% = 1: PDISO.ALARM% = 32: PDISO.CLEAN% = 64

  MAX.SAMPLES.FOR.EVENT% = MAX.WALK.TIME% * INTERRUPT.RATE
  SAMPLES.IN.RISETIME% = RISETIME * INTERRUPT.RATE

  CALQUEST$(1) = "Enter the person's last name."
  CALQUEST$(2) = "Enter the person's first name."
  CALQUEST$(3) = "Enter the person's middle initial."
  CALQUEST$(4) = "Enter the person's identification number - use 8 numerals."
  CALQUEST$(5) = "Enter the person's sex.      M-male      F-female"
  CALQUEST$(6) = "Enter the person's height - use centimetres."
  CALQUEST$(7) = "Enter the person's weight - use kilograms."
  CALQUEST$(8) = "Does the person wear a dental bridge?    Y-yes    N-no"
  CALQUEST$(9) = "Does the person have a metallic implant ?     Y-yes     N-no"
  CALQUEST$(10) = "Does the person have a pacemaker?     Y-yes     N-no"
  CALQUEST$(11) = "Enter any other features beside the 'other' heading "
  CALQUEST$(11) = CALQUEST$(11) + "- otherwise enter 'NONE'"
  TIME.TEST% = 0: JEND% = 1: IEND% = 20000
  WHILE TIME.TEST% = 0
    JEND% = 2 * JEND%
    STTIME = TIMER
    FOR J% = 1 TO JEND%: FOR I% = 1 TO IEND%: NEXT I%: NEXT J%
    ENDTIME = TIMER
    IF ENDTIME - STTIME > 1.5 THEN TIME.TEST% = 1
  WEND
  LOOPSPERSEC = JEND% / (ENDTIME - STTIME) * IEND%
  BLINKTIME = .05
  TIMELOOP% = BLINKTIME * LOOPSPERSEC

  MAX.PASSNUM% = 8
  NUM.FEATURES% = 10
  NUM.OPERATE.ALERTS% = 8

  OP.NSTDEV(1) = 4!: CAL.NSTDEV(1) = 3!: LIM.STDEV(1) = 4: '     DEL-X
  OP.NSTDEV(2) = 3!: CAL.NSTDEV(2) = 2!: LIM.STDEV(2) = 3: '     DEL-Y
```

```
  OP.NSTDEV(3) = 3!: CAL.NSTDEV(3) = 2!: LIM.STDEV(3) = .5: '    DIF.AN
  OP.NSTDEV(4) = 3!: CAL.NSTDEV(4) = 2!: LIM.STDEV(4) = 1: '    MID.AN
  OP.NSTDEV(5) = 4!: CAL.NSTDEV(5) = 3!: LIM.STDEV(5) = 2: '    ANGL1
  OP.NSTDEV(6) = 4!: CAL.NSTDEV(6) = 3!: LIM.STDEV(6) = 2: '    ANGL2
  OP.NSTDEV(7) = 4!: CAL.NSTDEV(7) = 3!: LIM.STDEV(7) = 2: '    ANGR1
  OP.NSTDEV(8) = 4!: CAL.NSTDEV(8) = 3!: LIM.STDEV(8) = 2: '    ANGR2
  OP.NSTDEV(9) = 4!: CAL.NSTDEV(9) = 3!: LIM.STDEV(9) = 2: '    DELYL
  OP.NSTDEV(10) = 4!: CAL.NSTDEV(10) = 3!: LIM.STDEV(10) = 2: '    DELYR
  RECOMP.CHECK% = 2 * X.DEL.CHK.S%
  RETURN

END SUB

SUB PMDHEAD
'  This subprogram calls all the heading and data entry positions.
'  The data statements are contained in the main module.
  DIM HEADR%(25), HEADC%(25), HEADT$(25)
  DIM DATASPCS%(25)

  OPEN "C:\PMD\DATA\HEADING1.DAT" FOR INPUT AS #1
  GOSUB CALIBRATE.HEADING
  GOSUB UPDATE.HEADING
  GOSUB UPDATE.ADD.HEADING
  GOSUB CONSTANT.HEADING
  GOSUB CALIBRATE.POSN
  GOSUB UPDATE.POSN
  GOSUB CONSTANT.POSN
  CLOSE #1
  EXIT SUB

GENERAL.HEADING: ' - routine to set up a general heading
  A$ = ""
  WHILE LEFT$(A$, 3) <> "998"
    LINE INPUT #1, A$
    PRINT A$
  WEND
  IHEAD% = 0
  INPUT #1, IROW%
  WHILE IROW% <> 999
    INPUT #1, NHEAD%
    FOR I% = 1 TO NHEAD%
      INPUT #1, ICOL%, HEAD$
      IHEAD% = IHEAD% + 1
      HEADR%(IHEAD%) = IROW%: HEADC%(IHEAD%) = ICOL%
      HEADT$(IHEAD%) = HEAD$
      NEXT I%
    INPUT #1, IROW%
    WEND
  NUMBER.OF.HEADINGS% = IHEAD%
  RETURN

CALIBRATE.HEADING: ' - routine to set up the CALIBRATE screen
```

```
   GOSUB GENERAL.HEADING
   NUM.CAL.HEAD% = NUMBER.OF.HEADINGS%
   FOR I% = 1 TO NUM.CAL.HEAD%
     CALHEADR%(I%) = HEADR%(I%)
     CALHEADC%(I%) = HEADC%(I%)
     CALHEADT$(I%) = HEADT$(I%)
     NEXT I%
   RETURN

UPDATE.HEADING: ' - routine to set up the UPDATE screen
   GOSUB GENERAL.HEADING
   NUM.UPDATE.HEAD% = NUMBER.OF.HEADINGS%
   FOR I% = 1 TO NUM.UPDATE.HEAD%
     UPDATEHEADR%(I%) = HEADR%(I%)
     UPDATEHEADC%(I%) = HEADC%(I%)
     UPDATEHEADT$(I%) = HEADT$(I%)
     NEXT I%
   RETURN


UPDATE.ADD.HEADING: ' - routine to set up the UPDATE.ADD screen
   GOSUB GENERAL.HEADING
   NUM.UPDATE.ADD.HEAD% = NUMBER.OF.HEADINGS%
   FOR I% = 1 TO NUM.UPDATE.ADD.HEAD%
     UPDATEADDHEADR%(I%) = HEADR%(I%)
     UPDATEADDHEADC%(I%) = HEADC%(I%)
     UPDATEADDHEADT$(I%) = HEADT$(I%)
     NEXT I%
   RETURN


CONSTANT.HEADING: ' - routine to set up the CONSTANT screen
   GOSUB GENERAL.HEADING
   NUM.CON.HEAD% = NUMBER.OF.HEADINGS%
   FOR I% = 1 TO NUM.CON.HEAD%
     CONHEADR%(I%) = HEADR%(I%)
     CONHEADC%(I%) = HEADC%(I%)
     CONHEADT$(I%) = HEADT$(I%)
     NEXT I%
   RETURN


GENERAL.DATA.POSN: '    -    routine for entering data entry positions
   A$ = ""
   WHILE LEFT$(A$, 3) <> "998"
     LINE INPUT #1, A$
     PRINT A$
   WEND
   IHEAD% = 0
   INPUT #1, IROW%
   WHILE IROW% <> 999
     INPUT #1, NHEAD%
```

```
    FOR I% = 1 TO NHEAD%
      INPUT #1, ICOL%, ISPC%
      IHEAD% = IHEAD% + 1
      DATAR%(IHEAD%) = IROW%: DATAC%(IHEAD%) = ICOL%: DATASPCS%(IHEAD%) = ISPC%
      NEXT I%
    INPUT #1, IROW%
    WEND
  NUMBER.OF.POSITIONS% = IHEAD%
  RETURN

CALIBRATE.POSN: ' - routine for entering CALIBRATE data positions
  GOSUB GENERAL.DATA.POSN
  NUM.CAL.PARM% = NUMBER.OF.POSITIONS%
  FOR I% = 1 TO NUMBER.OF.POSITIONS%
    CALROW%(I%) = DATAR%(I%)
    CALCOL%(I%) = DATAC%(I%)
    CALSPCS%(I%) = DATASPCS%(I%)
    NEXT I%
  RETURN


UPDATE.POSN: ' - routine for entering UPDATE.ADD data positions
  GOSUB GENERAL.DATA.POSN
  NUM.UPDATE.PARM% = NUMBER.OF.POSITIONS%
  FOR I% = 1 TO NUMBER.OF.POSITIONS%
    UPDATEROW%(I%) = DATAR%(I%)
    UPDATECOL%(I%) = DATAC%(I%)
    UPDATESPCS%(I%) = DATASPCS%(I%)
    NEXT I%
  RETURN


CONSTANT.POSN: ' - routine for entering CONSTANT data positions
  GOSUB GENERAL.DATA.POSN
  NUM.CON.PARM% = NUMBER.OF.POSITIONS%
  FOR I% = 1 TO NUMBER.OF.POSITIONS%
    CONROW%(I%) = DATAR%(I%)
    CONCOL%(I%) = DATAC%(I%)
    CONSPCS%(I%) = DATASPCS%(I%)
    NEXT I%
  RETURN

END SUB

SUB READ.KEYBOARD
' - routine to read the keyboard to get the employee number
SHARED PERSON.WAITING$, NEED.TO.BLANK$, F1HIT$
SHARED LASTKEY$, EMPNO%, DEMPNO%(), NUMBER.OF.EMPLOYEES%
SHARED INDEX.NUM%
SHARED YES$, NO$, OK$, NOTOK$
  PERSON.WAITING$ = NO$
  VIEW PRINT 22 TO 24
```

```
  IF NEED.TO.BLANK$ = YES$ THEN
    CLS 2
    CALL BLANKLINE(23): PRINT "Press any key to initiate a test.";
    SOUND 100, 3: SOUND 50, 2
    NEED.TO.BLANK$ = NO$
  END IF
  A$ = LASTKEY$

  IF LEN(A$) <> 0 THEN
    CALL BLANKLINE(23): SOUND 500, 2: SOUND 200, 2
    INPUT ; "Enter employee number : ", EMPNO%
    NEED.TO.BLANK$ = YES$
    IF EMPNO% > 0 THEN : PERSON.WAITING$ = YES$: CALL BLANKLINE(23)
  END IF
  VIEW PRINT
  EXIT SUB
END SUB

SUB SWAP.EMPLOYEE (HIGH.INDEX.NUM%, ARRAY.ENTRY.POINT%)
' - routine to swap the employee at the proper location in the database.
  IF ARRAY.ENTRY.POINT% < NUMBER.OF.EMPLOYEES% THEN
    FOR J% = HIGH.INDEX.NUM% TO ARRAY.ENTRY.POINT% + 1 STEP -1
      JM% = J% - 1
      SWAP DEMPNO%(J%), DEMPNO%(JM%)
      SWAP DACTLAST$(J%), DACTLAST$(JM%)
      SWAP DACTFIRST$(J%), DACTFIRST$(JM%)
      SWAP DACTINIT$(J%), DACTINIT$(JM%)
      SWAP DSEX(J%), DSEX(JM%): SWAP DHEIGHT(J%), DHEIGHT(JM%)
      SWAP DWEIGHT(J%), DWEIGHT(JM%): SWAP DBRIDGE(J%), DBRIDGE(JM%)
      SWAP DIMPLANT(J%), DIMPLANT(JM%): SWAP DPACE(J%), DPACE(JM%)
      SWAP DOTHER$(J%), DOTHER$(JM%)
      FOR I% = 1 TO NUM.FEATURES%
        SWAP DMEAN(J%, I%), DMEAN(JM%, I%)
        SWAP DSTDEV(J%, I%), DSTDEV(JM%, I%)
      NEXT I%
    NEXT J%
  END IF
  IN.SECURE.AREA%(ARRAY.ENTRY.POINT%) = 0
  EXIT SUB
END SUB

SUB UPDATE.EMPLOYEE.DATABASE
SHARED INDEX.NUM%
' - routine to update the employee database
  DIM UPDATEPARM$(5)
  RESPONSE$ = ""
  VALID.ENTRY$ = "YES"
  WHILE RESPONSE$ <> "Q" AND RESPONSE$ <> "q"
    GOSUB UPDATE.HEAD.DISPLAY
    IF VALID.ENTRY$ = "NO" THEN
      LOCATE 21, 1: PRINT "Invalid entry - try again  ";
      SOUND 500, 2: SOUND 300, 2: VALID.ENTRY$ = "YES"
```

```
    END IF
    LOCATE 7, 54: PRINT NUMBER.OF.EMPLOYEES%
    LOCATE 19, 2: PRINT "Enter your selection ( A, D, C,or Q ) ";
    SOUND 100, 3: SOUND 50, 2
    CALL GETINKEY
    RESPONSE$ = A$
    PRINT A$;
    SELECT CASE RESPONSE$
    CASE "A", "a"
      GOSUB ADD.EMPLOYEE
    CASE IS = "D", "d"
      GOSUB DELETE.EMPLOYEE
    CASE "C", "c"
      GOSUB CHANGE.EMPLOYEE
    CASE IS = "Q", "q"
      LOCATE 23, 1: PRINT "STOP"
    CASE ELSE
      VALID.ENTRY$ = "NO"
    END SELECT
  WEND
  EXIT SUB

UPDATE.HEAD.DISPLAY: ' - routine to display the UPDATE screen
  SCREEN 0: CLS : VIEW PRINT
  FOR I% = 1 TO NUM.UPDATE.HEAD%
    LOCATE UPDATEHEADR%(I%), UPDATEHEADC%(I%)
    PRINT UPDATEHEADT$(I%);
  NEXT I%
  RETURN

ADD.EMPLOYEE: ' - routine to add an employee to the database
  VIEW PRINT 9 TO 24
  CLS 2
  LOCATE 25, 1: COLOR 10, 1: PRINT "Press 'Esc' key to exit.": COLOR 2, 1
  LOCATE 9, 1
  EXIT$ = "NO"
  ALLDATA$ = "NOT OK"
  IPRMMAX% = NUM.UPDATE.PARM%
  FOR I% = 1 TO IPRMMAX%
    DATAR%(I%) = UPDATEROW%(I%): DATAC%(I%) = UPDATECOL%(I%)
    SPCS%(I%) = UPDATESPCS%(I%)
  NEXT I%
  ANOTHER$ = "Y"
  WHILE ANOTHER$ = "Y"
    IPRM% = 1
    GOSUB UPDATE.ADD.HEAD.DISPLAY
    ALLDATA$ = "NOT OK"
    WHILE ALLDATA$ = "NOT OK"
      FINISH$ = "NO"
      WHILE FINISH$ = "NO"
      IF IPRM% > IPRMMAX% THEN IPRM% = 1
      IF IPRM% <= 0 THEN IPRM% = IPRMMAX%
```

```
      SOUND 100, 3: SOUND 50, 2
      ROW% = DATAR%(IPRM%): COLUMN% = DATAC%(IPRM%)
      LOCATE ROW%, COLUMN%
      CALL ADDNEWCHAR
      IF EXIT$ = "YES" THEN RETURN
      CALL BLANKLINE(18): CALL BLANKLINE(17)
    WEND
    GOSUB UPDATE.CHECK
    IF ALLDATA$ = "OK" THEN
      GOSUB UPDATE.EMP.NUM.IN.USE.CHECK
      IF EMP.NUM.IN.USE$ = "YES" THEN
        CALL BLANKLINE(18): CALL BLANKLINE(17)
        PRINT "This employee number is currently in use - ";
        PRINT "enter a new value"
        PRINT " or delete the other employee from the database."
        SOUND 500, 2: SOUND 300, 2
        IPRM% = 4: ALLDATA$ = "NOT OK"
      END IF
    END IF
    IF ALLDATA$ = "OK" THEN
      ARRAY.ENTRY.POINT% = NUMBER.OF.EMPLOYEES% + 1
      GOSUB UPDATE.ADD.DATABASE
      FOR J% = 1 TO NUM.FEATURES%
        DMEAN(ARRAY.ENTRY.POINT%, J%) = 0
        DSTDEV(ARRAY.ENTRY.POINT%, J%) = 0
      NEXT J%
      CALL BLANKLINE(17): CALL BLANKLINE(18)
      PRINT "Enter another?  Y(yes)   N(no) ";
      CALL GETINKEY
      ANOTHER$ = UCASE$(A$)
    END IF
  WEND
  RETURN

UPDATE.ADD.HEAD.DISPLAY: ' - routine to display the UPDATE.ADD screen
  SCREEN 0: CLS : VIEW PRINT
  FOR I% = 1 TO NUM.UPDATE.ADD.HEAD%
    LOCATE UPDATEADDHEADR%(I%), UPDATEADDHEADC%(I%)
    PRINT UPDATEADDHEADT$(I%);
  NEXT I%
  FOR I% = 1 TO NUM.UPDATE.PARM%
    LOCATE UPDATEROW%(I%), UPDATECOL%(I%)
    PRINT LEFT$(SPACECHR$, UPDATESPCS%(I%));
  NEXT I%
  RETURN

UPDATE.CHG.HEAD.DISPLAY: ' - routine to display the UPDATE.CHG screen
  SCREEN 0: VIEW PRINT: CLS
  FOR I% = 1 TO NUM.UPDATE.ADD.HEAD%
    LOCATE UPDATEADDHEADR%(I%), UPDATEADDHEADC%(I%)
    PRINT UPDATEADDHEADT$(I%);
  NEXT I%
```

```
  FOR I% = 1 TO NUM.UPDATE.PARM%
    LOCATE UPDATEROW%(I%), UPDATECOL%(I%)
    PRINT LEFT$(SPACECHR$, UPDATESPCS%(I%));
  NEXT I%
  LOCATE UPDATEROW%(1), UPDATECOL%(1): PRINT DACTLAST$(ECHG%)
  LOCATE UPDATEROW%(2), UPDATECOL%(2): PRINT DACTFIRST$(ECHG%)
  LOCATE UPDATEROW%(3), UPDATECOL%(3): PRINT DACTINIT$(ECHG%)
  LOCATE UPDATEROW%(4), UPDATECOL%(4) - 1: PRINT DEMPNO%(ECHG%)
  RETURN

DELETE.EMPLOYEE: ' - routine to remove an employee from the database.
  VIEW PRINT 7 TO 25
  CLS 2
  LOCATE 8, 10
  INPUT "Enter the  employee number to delete from the database "; AEMPNO%
  EMP.FOUND$ = "NO"
  FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
    IF AEMPNO% = DEMPNO%(I%) THEN
      EMP.FOUND$ = "YES"
      ARRAY.ENTRY.POINT% = I%
      I% = NUMBER.OF.EMPLOYEES% + 1
    END IF
  NEXT I%
  IF EMP.FOUND$ = "YES" THEN
    I% = ARRAY.ENTRY.POINT%
    ANAME$ = DACTFIRST$(I%) + " " + DACTINIT$(I%) + " " + DACTLAST$(I%)
    LOCATE 10, 5: PRINT "Do you wish to delete -  ";
    PRINT ANAME$;
    INPUT " ?     Y(Yes)  N(No) ", ANS$
    IF ANS$ = "Y" OR ANS$ = "y" THEN
      INDEX.NUM% = I%
      CALL DELETE.EMPLOYEE.FROM.DATABASE
      LPRINT "Employee "; ANAME$; " number "; AEMPNO%; " deleted."
    END IF
  ELSE
    LOCATE 10, 10: PRINT "Employee number "; AEMPNO%; "  is not found."
    LOCATE 12, 10: PRINT "Press any key to continue ";
    CALL GETINKEY
  END IF
  VIEW PRINT
  RETURN

CHANGE.EMPLOYEE: ' - routine to change an employee's data in the database.
'  This routine is to be used when an existing employee changes either
'  their name or employee number.
  VIEW PRINT 7 TO 25: CLS 2: LOCATE 8, 10
  INPUT "Enter the employee's last name : ", CHGLASTNAME$
  LOCATE , 10: INPUT "Enter the employee's number : ", CHGNUMBER%
  EMP.FOUND$ = "NO"
  FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
    IF CHGNUMBER% = DEMPNO%(I%) THEN
    ECHG% = I%
```

```
      I% = ECHG% + 1
      EMP.FOUND$ = "YES"
    END IF
  NEXT I%
  IF EMP.FOUND$ = "NO" THEN
    PRINT "An entry in the database with Employee # ";
    PRINT CHGNUMBER%; " is not found."
    INPUT "    - press return to continue.", A$
    RETURN
  END IF
  IF DACTLAST$(ECHG%) <> CHGLASTNAME$ THEN
    PRINT "Last names do not match - employee # "; CHGNUMBER%; "  is ";
    PRINT DACTLAST$(ECHG%): INPUT "     ; press return to continue", A$
    RETURN
  END IF
' The employee number and last names are okay - now ready to accept changes.
  ALLDATA$ = "NOT OK"
  IPRMMAX% = NUM.UPDATE.PARM%
  FOR I% = 1 TO IPRMMAX%
    DATAR%(I%) = UPDATEROW%(I%): DATAC%(I%) = UPDATECOL%(I%)
    SPCS%(I%) = UPDATESPCS%(I%)
  NEXT I%
  ANOTHER$ = "Y"
  IPRM% = 1
  GOSUB UPDATE.CHG.HEAD.DISPLAY
  ALLDATA$ = "NOT OK"
  WHILE ALLDATA$ = "NOT OK"
    FINISH$ = "NO"
    WHILE FINISH$ = "NO"
      IF IPRM% > IPRMMAX% THEN IPRM% = 1
      IF IPRM% <= 0 THEN IPRM% = IPRMMAX%
      SOUND 100, 3: SOUND 50, 2
      ROW% = DATAR%(IPRM%): COLUMN% = DATAC%(IPRM%)
      LOCATE ROW%, COLUMN%
      CALL ADDNEWCHAR
      IF EXIT$ = "YES" THEN RETURN
      CALL BLANKLINE(17)
    WEND
    GOSUB UPDATE.CHECK
    GOSUB UPDATE.EMP.NUM.IN.USE.CHECK
    IF EMP.NUM.IN.USE$ = "YES" AND IN.USE.NUM% <> ECHG% THEN
      CALL BLANKLINE(17)
      PRINT "This employee number is currently in use - ";
      PRINT "enter a new value."
      SOUND 500, 2: SOUND 300, 2
      IPRM% = 4: ALLDATA$ = "NOT OK"
    END IF
    IF ALLDATA$ = "OK" THEN
      GOSUB UPDATE.CHECK
      DACTLAST$(ECHG%) = UPDATEPARM$(1)
      DACTFIRST$(ECHG%) = UPDATEPARM$(2)
      DACTINIT$(ECHG%) = UPDATEPARM$(3)
```

```
      DEMPNO%(ECHG%) = VAL(UPDATEPARM$(4))
      DATA.CHANGE$ = "YES"
    END IF
  WEND
  RETURN

UPDATE.CHECK: ' - routine to check that the UPDATE values are ok, and to
'       place values in the UPDATEPARM$() array.  This routine is called
'       from ADD.EMPLOYEE and CHANGE.EMPLOYEE.
  ALLDATA$ = "OK"
  FOR I% = 1 TO NUM.UPDATE.PARM%
    B$ = " "
    FOR J% = 1 TO UPDATESPCS%(I%)
      A$ = CHR$(SCREEN(UPDATEROW%(I%), UPDATECOL%(I%) + J% - 1))
      IF A$ = ASPACECHR$ THEN
        J% = UPDATESPCS%(I%) + 1
      ELSE
        B$ = B$ + A$
      END IF
    NEXT J%
    UPDATEPARM$(I%) = RTRIM$(LTRIM$(B$))
  NEXT I%
  FOR I% = 1 TO NUM.UPDATE.PARM%
    IF LEN(UPDATEPARM$(I%)) < 1 AND I% <> 3 THEN
      ALLDATA$ = "NOT OK"
      CALL BLANKLINE(17): PRINT "A value must be entered here."
      SOUND 500, 2: SOUND 300, 2
      IPRM% = I%
      RETURN
    END IF
  NEXT I%
  AEMPNO% = VAL(UPDATEPARM$(4))
  IF AEMPNO% <= 0 THEN
    ALLDATA$ = "NOT OK"
    CALL BLANKLINE(17): PRINT "This employee # is not valid - ";
    PRINT "enter a new value."
    SOUND 500, 2: SOUND 300, 2
    IPRM% = 4
    RETURN
  END IF
  RETURN

UPDATE.EMP.NUM.IN.USE.CHECK: ' - routine to determine if the employee # is in
use
  EMP.NUM.IN.USE$ = "NO"
  FOR I% = 1 TO NUMBER.OF.EMPLOYEES%
    IF DEMPNO%(I%) = AEMPNO% THEN
      IN.USE.NUM% = I%
      I% = NUMBER.OF.EMPLOYEES% + 1
      EMP.NUM.IN.USE$ = "YES"
      RETURN
    END IF
```

```
  NEXT I%
  RETURN

UPDATE.ADD.DATABASE: ' - routine to UPDATE the database
  K% = ARRAY.ENTRY.POINT%
  DEMPNO%(K%) = AEMPNO%
  DACTLAST$(K%) = UPDATEPARM$(1)
  DACTFIRST$(K%) = UPDATEPARM$(2)
  DACTINIT$(K%) = UPDATEPARM$(3)
  DATA.CHANGE$ = "YES"
  NUMBER.OF.EMPLOYEES% = NUMBER.OF.EMPLOYEES% + 1
  CALL BLANKLINE(24)
  LPRINT "Employee "; DACTFIRST$(K%); "  "; DACTINIT$(K%); " "; DACTLAST$(K%);
  LPRINT " number "; DEMPNO%(K%); " added."
  RETURN

END SUB
```

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OF PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A method of detecting concealed metals comprising the steps of:

generating a fluctuating electromagnetic field about a passageway;

passing a person or object through said passageway;

obtaining an electrical signal representative of said electromagnetic field as said person or object passes through said passageway;

comparing the values of predetermined characteristics of said electrical signal against the corresponding values of said characteristics of a base electrical signal for said person or object previously obtained when said person or object was known not to contain concealed metals; and

generating an alarm signal when the differences between at least one of said characteristics signals exceeds a predetermined threshold.

2. A method as defined in claim 1, said producing step including processing said electrical signal to determine quantitative values for each of said characteristics of said signal and said comparing step including comparing said quantitative values against corresponding base values of said characteristics for said person or object previously obtained when said person or object was known not to contain concealed metals.

3. A method as defined in claim 2, said processing step including demodulating said electrical signal to produce a first signal representative of the resistive component of said electromagnetic field and a second signal representative of the reactive component of said electromagnetic field.

4. A method as defined in claim 3, said processing step including the steps of:

determining the magnitude of each said first and second signals at predetermined increments of time;

determining the maximum amplitude of each of said first and second signals;

determining an overall differential phase angle between said first and second signals;

determining an intermediate differential phase angle between said first and second signals; and

determining a differential phase angle for each of a plurality of different positions of said person or object in said passageway.

5. An apparatus for detecting concealed metals on a person or object, comprising:

means for generating an electromagnetic field about a passageway;

means for producing an electrical signal representative of said electromagnetic field as said person or object passes through said passageway; and

means for comparing the values of predetermined characteristics of said signal against corresponding values representative of an electrical calibration signal obtained when said person or object was known not to contain concealed metals; and

means for generating an alarm signal when the differences between the values of at least one of said characteristics of said signals exceeds a predetermined threshold.

6. An apparatus as defined in claim 5, said producing means including means for demodulating said electrical signal to produce a first signal representative of the resistive component of said electromagnetic field and a second signal representative of the reactive component of said electromagnetic field.

7. An apparatus as defined in claim 6, said producing means being operable to determine the magnitude of each said first and second signals at predetermined increments of time, the maximum amplitude of each of said first and second signals, an overall differential phase angle between said first and second signals, an intermediate differential phase angle between said first and second signals, and a differential phase angle for each of a plurality of different positions of said person or object in said passageway.

8. An apparatus as defined in claim 5, said means for generating an electromagnetic field about a passageway including a transmit coil extending about said passageway with an axis extending longitudinally through said passageway and means for producing an alternating current at a predetermined frequency within said coil so as to form said magnetic field about said coil.

9. An apparatus as defined in claim 8, said means for producing an electrical signal including a pair of receive coils about said passageway and coaxially spaced apart on opposite sides of said transmit coil, said receive coils having a common terminal and output terminals and being responsive to a changing magnetic field within said passageway to produce alternating voltages across said receive coils.

10. An apparatus as defined in claim 9, further including an amplifier having input terminals connected to said output terminals of said receive coils and an output terminal for producing said electrical signal.

11. An apparatus as defined in claim 10, further including means for producing and matching a reference signal to said electrical signal from said receive coils and means for obtaining a difference signal which is the difference between the amplified receive coil output signal and said reference signal.

12. An apparatus as defined in claim 11, further including a demodulator for receiving said differential signal and producing first and second output signals representative of the resistive and reactive components of said differential signal.

13. An apparatus as defined in claim 12, further including converter means for converting first and second output signals to first and second digital output signals, respectively.

14. An apparatus as defined in claim 13, further including computation means for processing said first and second digital output signals to determine the values of predetermined characteristics of thereof and compare said values against predetermined calibration values of said characteristics representative of said person or object.

15. An apparatus as defined in claim 14, said computation means being operable to determine the magnitude of each said first and second signals at predetermined increments of time, the maximum amplitude of each of said first and second signals, an overall differential phase angle between said first and second signals, an intermediate differential phase angle between said first and second signals, and a differential phase angle and signal magnitudes for each of a plurality of different positions of said person or object in said passageway.

16. An apparatus as defined in claim 10, further including means for reducing the effect of objects moving in said passageway in close proximity to said receive coils.

17. An apparatus as defined in claim 16, said reducing means including field shaping coil means associated with said receive coil means and responding to the magnetic field in the vicinity thereof by inducing an additional magnetic field in said station whereby voltages induced in the receive coil means are substantially influenced by currents flowing in the associated field shaping coil means.

18. An apparatus as defined in claim 17, wherein further including means for maintaining said additional magnetic field in an selected out-of-phase relationship with respect to the magnetic field generated by said transmit coil means.

19. An apparatus as defined in claim 5, further including a portal unit defining said passageway through which the person moves with said coil means being coaxially located with respect to each other in planes generally perpendicular to the axis of the passageway along which the person moves.

20. An apparatus as defined in claim 18, said field shaping coil means comprising a pair of field shaping coils, each located adjacent to a respective one of said receive coils.

21. An apparatus as defined in claim 10, further including capacitor means connected to said common terminal to produce a resonant condition between said receive coils during operation such that as the voltage in one receive coil increases the voltage in the other decreases so the net voltage across both receive coils is substantially zero when there said passageway is empty.

22. An apparatus as defined in claim 10, further including means to zero or null any induced imbalanced voltages across the receive coils arising from coil winding inaccuracies and the like.

23. An apparatus as defined in any of the preceding claims, further including ferromagnetic shielding means located and arranged to reduce or eliminate eddy currents and resulting magnetic field disturbances arising in materials exterior to the detecting station.

24. An apparatus as defined in any of the preceding claims, further including field shielding coil means spaced outwardly from the receive coil and field shaping coil means and arranged such that currents induced in the field shielding coils by the fluctuating magnetic fields in the station create still further magnetic fields which act to reduce the strength of magnetic fields extending outwardly away from the detecting station.

25. For use in a metal detecting system wherein a person capable of carrying concealed metal pieces is caused to pass through a detecting station within which a rapidly fluctuating magnetic field exists and means are provided to sense changes in the magnetic field resulting from passage of the person's body and any metals carried thereon, the calibration method substantially as described with reference to Figure 6.

26. For use in a metal detecting system wherein a person capable of carrying concealed metal pieces is caused to pass through a detecting station within which a rapidly fluctuating magnetic field exists and means are provided to sense changes in the magnetic field resulting from passage of the person's body and any metals carried thereon, the detection method substantially as described with reference to Figure 7.

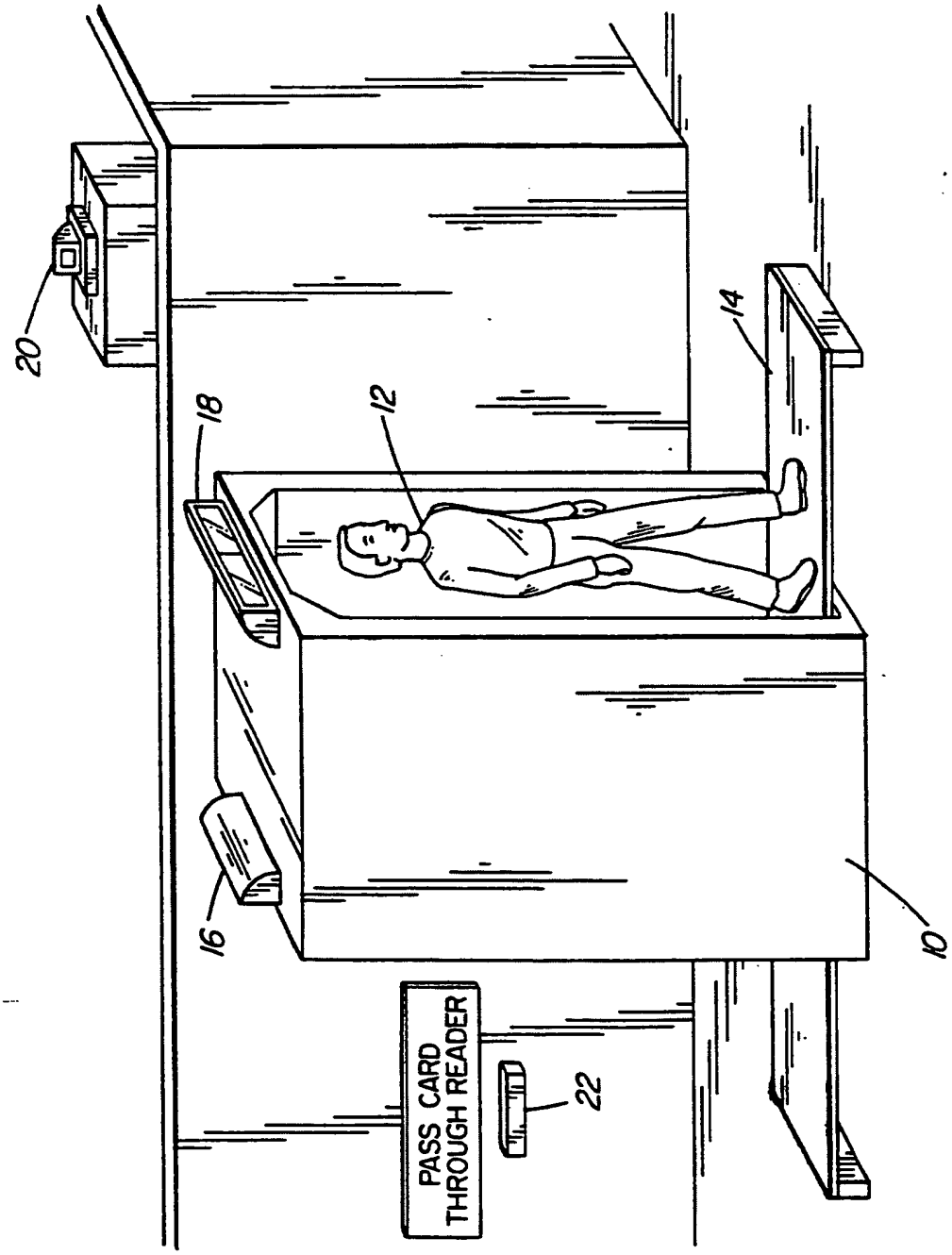


FIG. 1

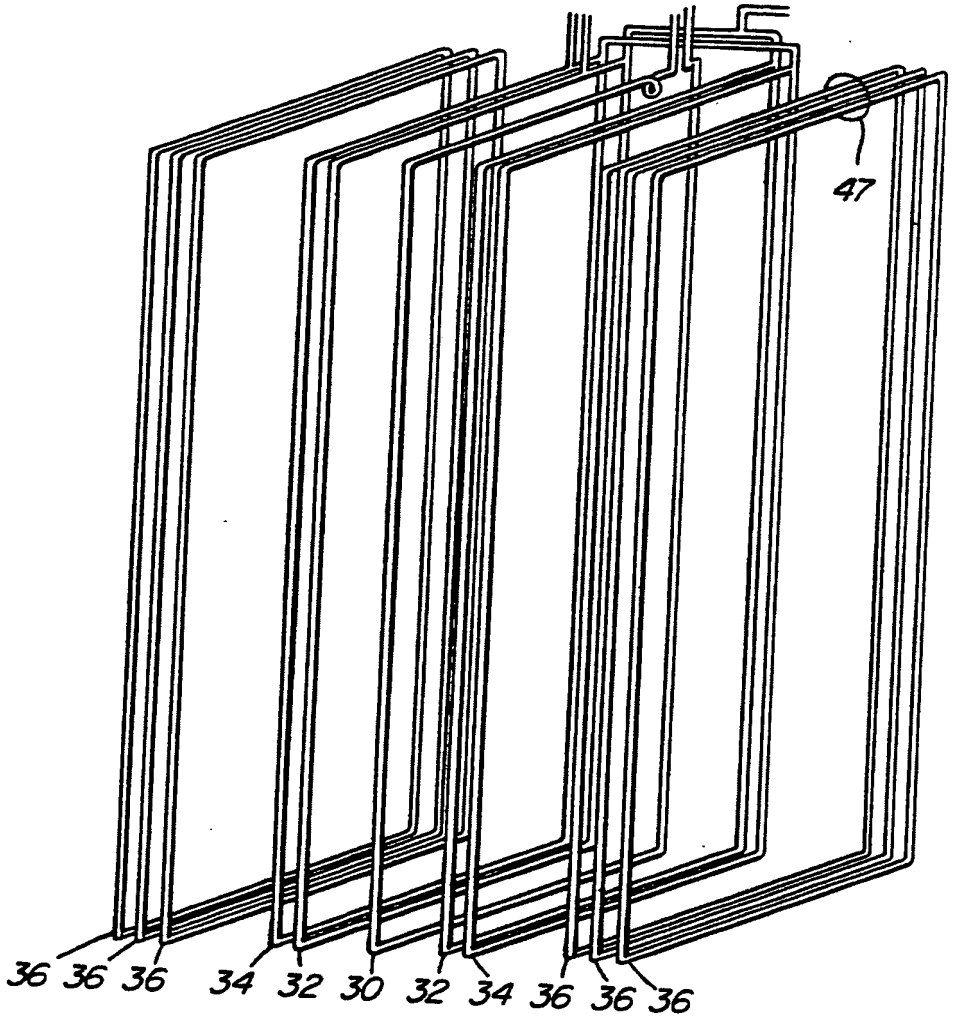


FIG. 2

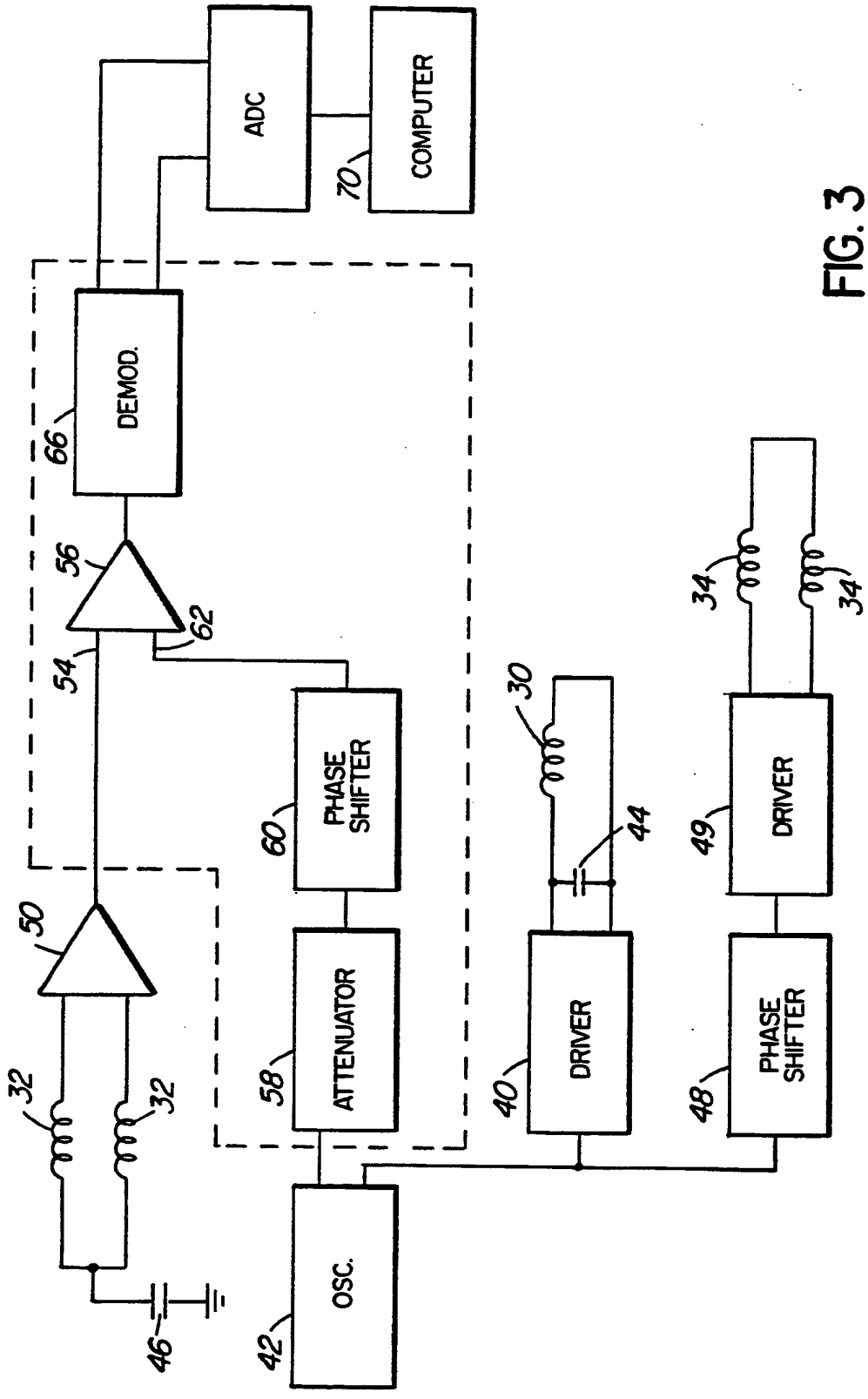

ADC
COMPUTER
70
DEMOD.
66
56
62
54
PHASE SHIFTER
60
30
44
34
34
50
ATTENUATOR
58
DRIVER
40
DRIVER
49
32
32
PHASE SHIFTER
48
46
OSC.
42

FIG. 3

FIG. 4a

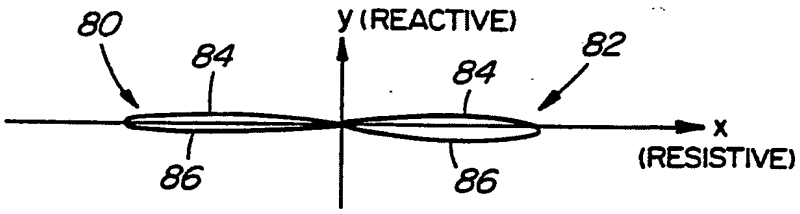


FIG. 4b

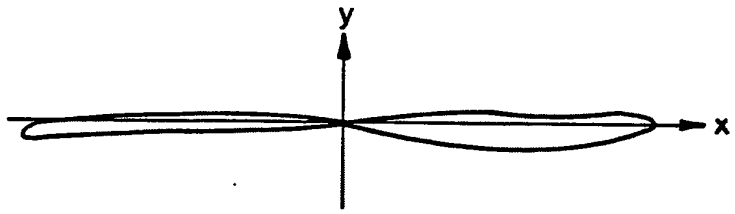


FIG. 4c

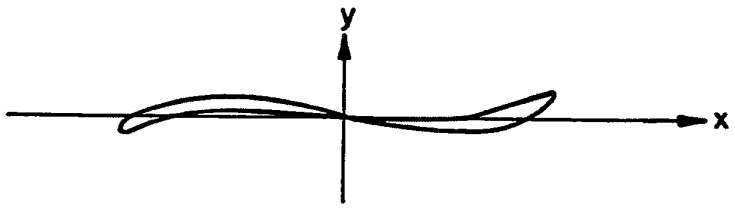


FIG. 4d

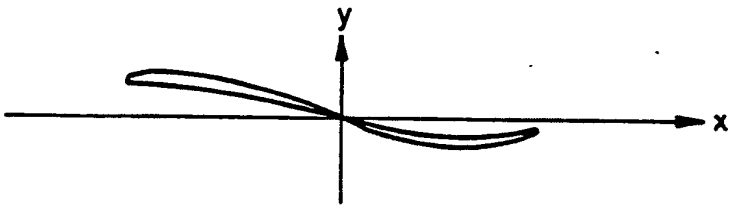


FIG. 4e

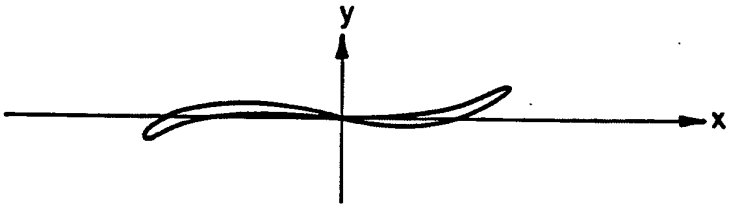
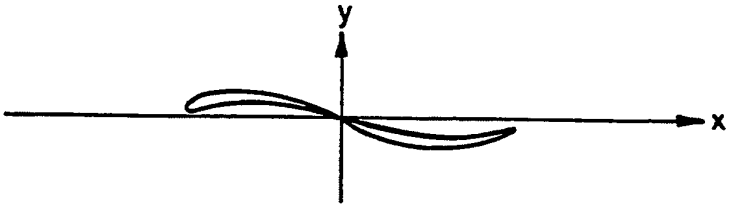


FIG. 4f



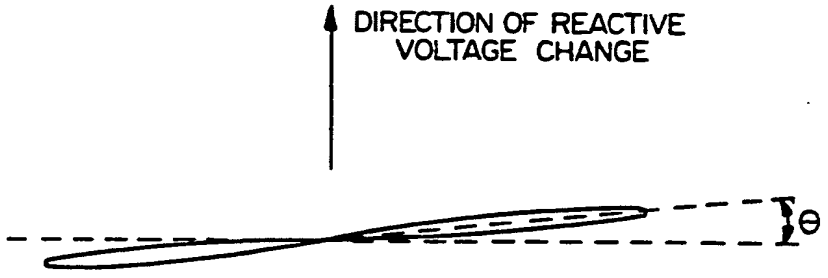


FIG. 5a

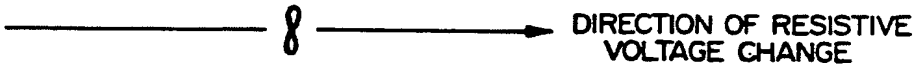


FIG. 5b

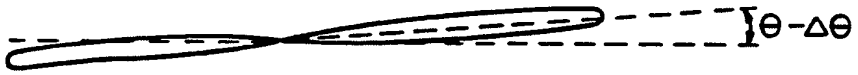


FIG. 5c

a-NORMAL BODY SIGNAL
b- SIGNAL FROM SMALL NON-FERROMAGNETIC OBJECT
c-COMBINED SIGNAL FROM THE NORMAL BODY AND THE SMALL NON-FERROMAGNETIC OBJECT

SIGNAL CHANGE IN RECEIVE COIL WITH DESIRED CURRENT IN FIELD-SHAPING COILS

NOTE THE SMALL SIGNAL FROM THE NON-FERROMAGNETIC OBJECT AND THE RESULTANT SMALL CHANGE IN ANGLE (Δθ) SEEN IN CURVE "c".

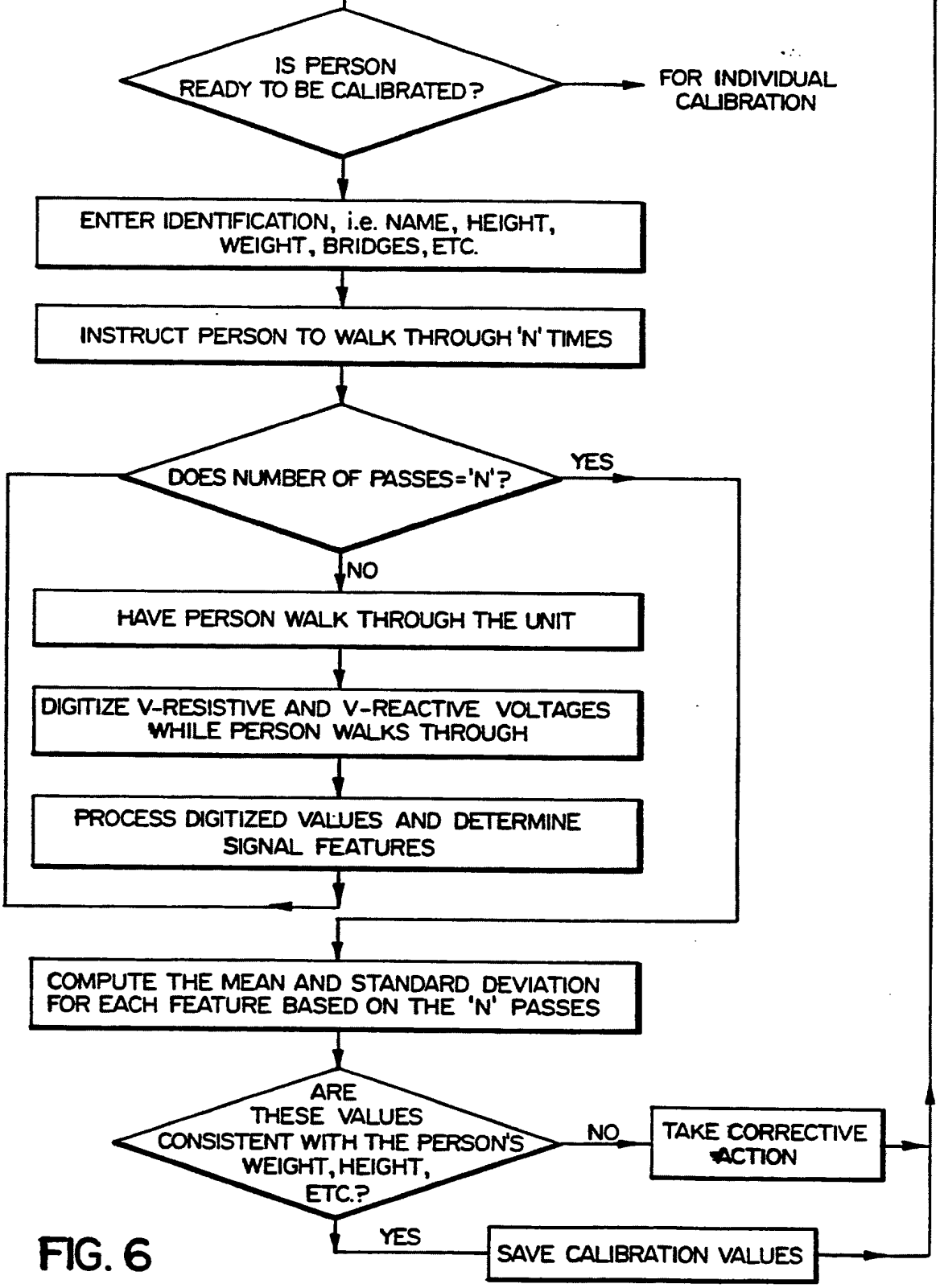

IS PERSON READY TO BE CALIBRATED?
FOR INDIVIDUAL CALIBRATION
ENTER IDENTIFICATION, i.e. NAME, HEIGHT, WEIGHT, BRIDGES, ETC.
INSTRUCT PERSON TO WALK THROUGH 'N' TIMES
DOES NUMBER OF PASSES='N'?
YES
NO
HAVE PERSON WALK THROUGH THE UNIT
DIGITIZE V-RESISTIVE AND V-REACTIVE VOLTAGES WHILE PERSON WALKS THROUGH
PROCESS DIGITIZED VALUES AND DETERMINE SIGNAL FEATURES
COMPUTE THE MEAN AND STANDARD DEVIATION FOR EACH FEATURE BASED ON THE 'N' PASSES
ARE THESE VALUES CONSISTENT WITH THE PERSON'S WEIGHT, HEIGHT, ETC.?
NO
TAKE CORRECTIVE ACTION
YES
SAVE CALIBRATION VALUES

FIG. 6

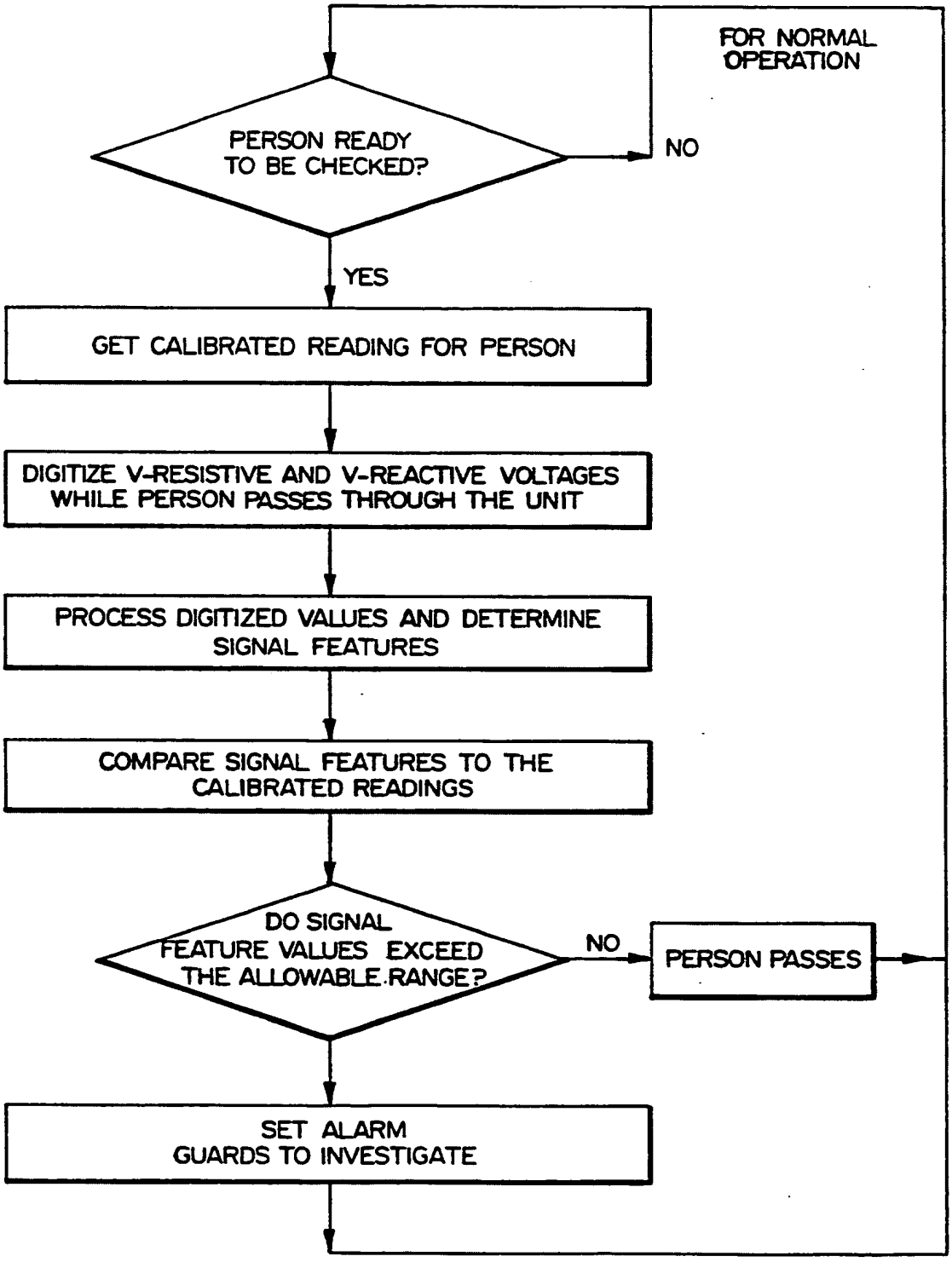


FIG. 7

SUBSTITUTE SHEET

INTERNATIONAL SEARCH REPORT

International Application No PCT/CA 90/00286

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G01V3/10

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G01V

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category[9]	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
A	US,A,4821023 (S.I.PARKS) 11 April 1989 see column 14, lines 51 - 63	1, 2, 16, 23
A	DE,A,2837265 (GEORGETOWN UNIVEWRSITY, WASHINGTON DC) 06 March 1980 see page 14, line 23 - page 15, line 6 see page 26, lines 1 - 8 see page 30, line 9 - page 31, line 15 see page 34, lines 17 - 25	1-10
A	EP,A,300974 (K.H.SCHMALL) 25 January 1989 see column 4, lines 12 - 26; figure 1	16, 17
A	EP,A,308073 (ANRITSU CORPORATION) 22 March 1989 see column 3, line 45 - column 6, line 27 see column 6, line 58 - column 7	3-15

[*] Special categories of cited documents:[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
03 DECEMBER 1990	19 DEC 1990
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	SWARTJES H.M.

Form PCT/ISA/210 (second sheet) (January 1985)

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. PCT/CA 90/00286

SA 39738

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on 03/12/90
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
US-A-4821023	11-04-89	EP-A-	0323713	12-07-89
DE-A-2837265	06-03-80	CA-A-	1157096	15-11-83
		CH-A-	660799	15-06-87
		FR-A,B	2456951	12-12-80
		GB-A-	1603578	25-11-81
		NL-A-	7903965	25-11-80
		SE-B-	440832	19-08-85
		SE-A-	7904329	18-11-80
EP-A-300974	25-01-89	None		
EP-A-308073	22-03-89	JP-A-	1054281	01-03-89
		US-A-	4949037	14-08-90